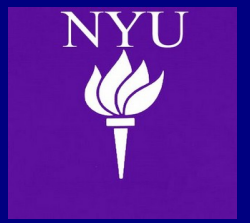
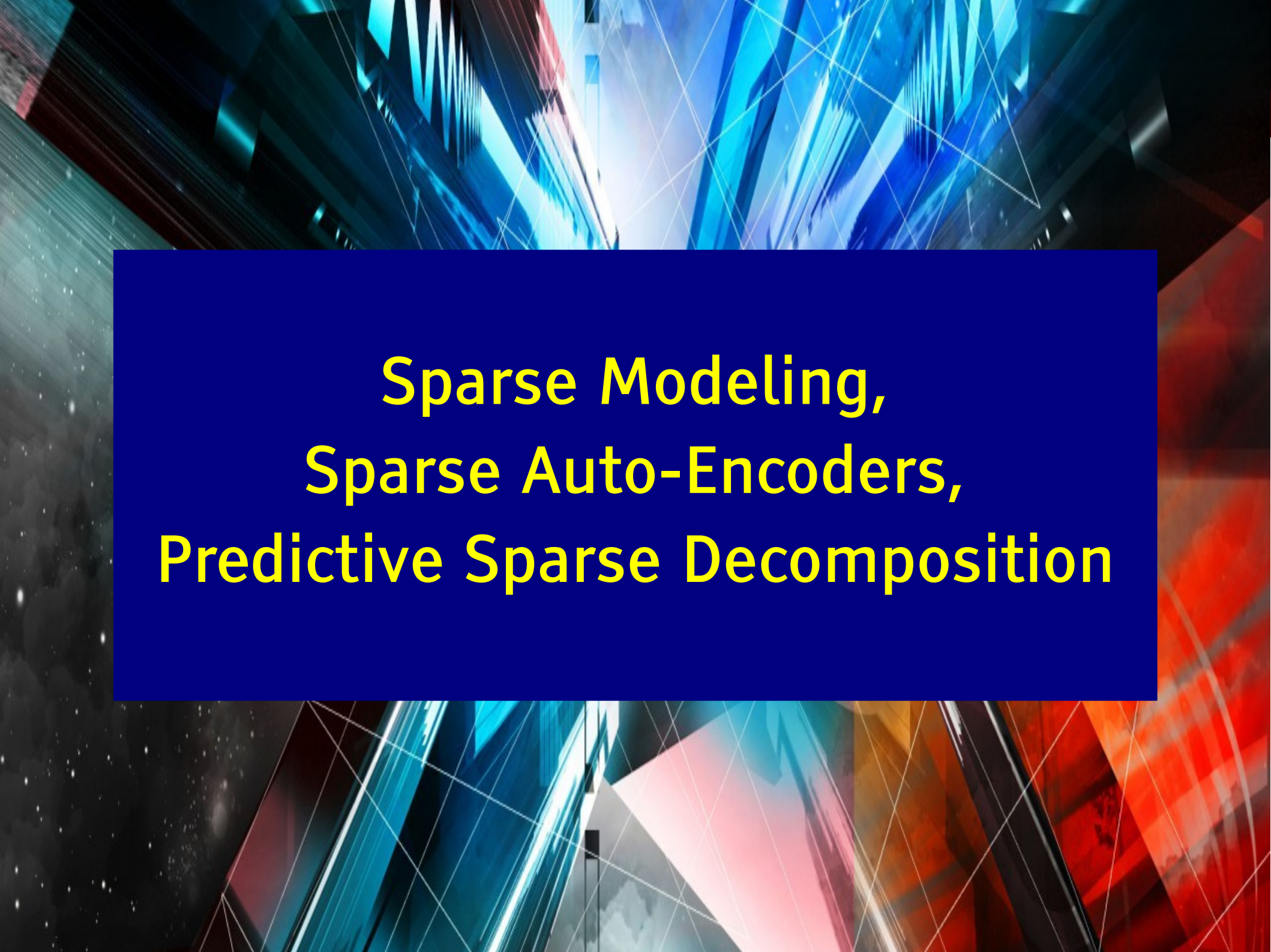
Unsupervised Learning part 2

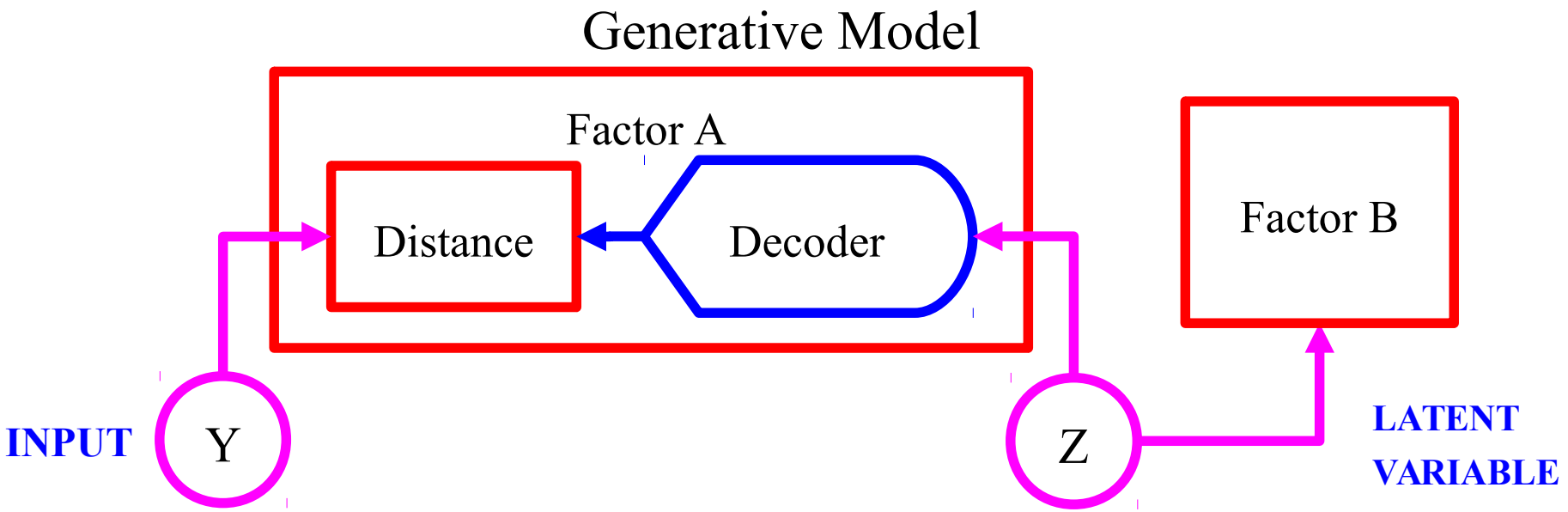
Yann Le Cun
Facebook AI Research,
Center for Data Science, NYU
Courant Institute of Mathematical Sciences, NYU
http://yann.lecun.com




Sparse Modeling,
Sparse Auto-Encoders,
Predictive Sparse Decomposition

How to Speed Up Inference in a Generative Model?

- Factor Graph with an asymmetric factor
- Inference Z → Y is easy
  - Run Z through deterministic decoder, and sample Y
- Inference Y → Z is hard, particularly if Decoder function is many-to-one
  - MAP: minimize sum of two factors with respect to Z
  - Z* = argmin_z Distance[Decoder(Z), Y] + FactorB(Z)
- Examples: K-Means (1of K), Sparse Coding (sparse), Factor Analysis

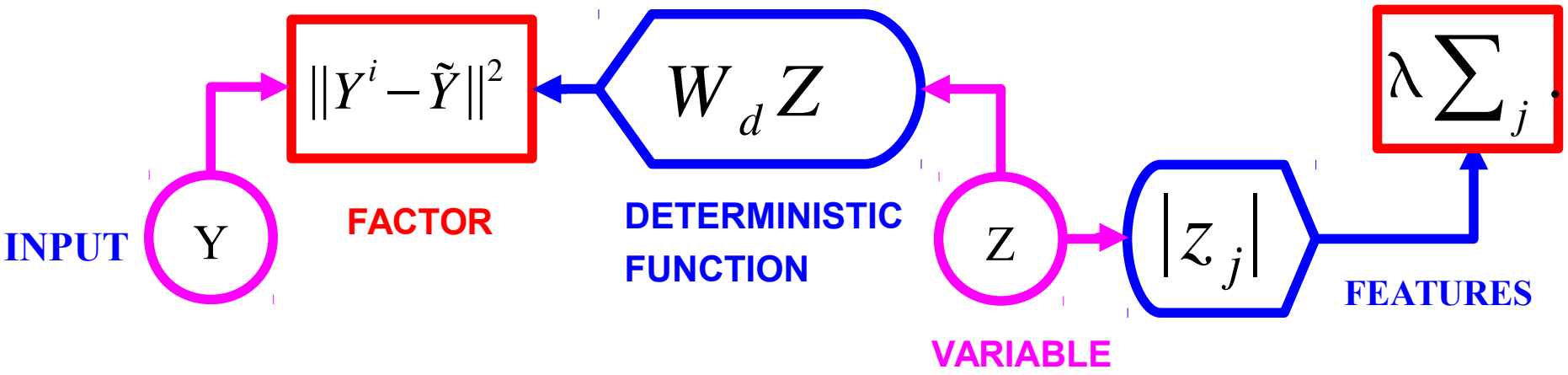


Sparse Coding & Sparse Modeling

[Olshausen & Field 1997]

- Sparse linear reconstruction
- Energy = reconstruction_error + code_prediction_error + code_sparsity

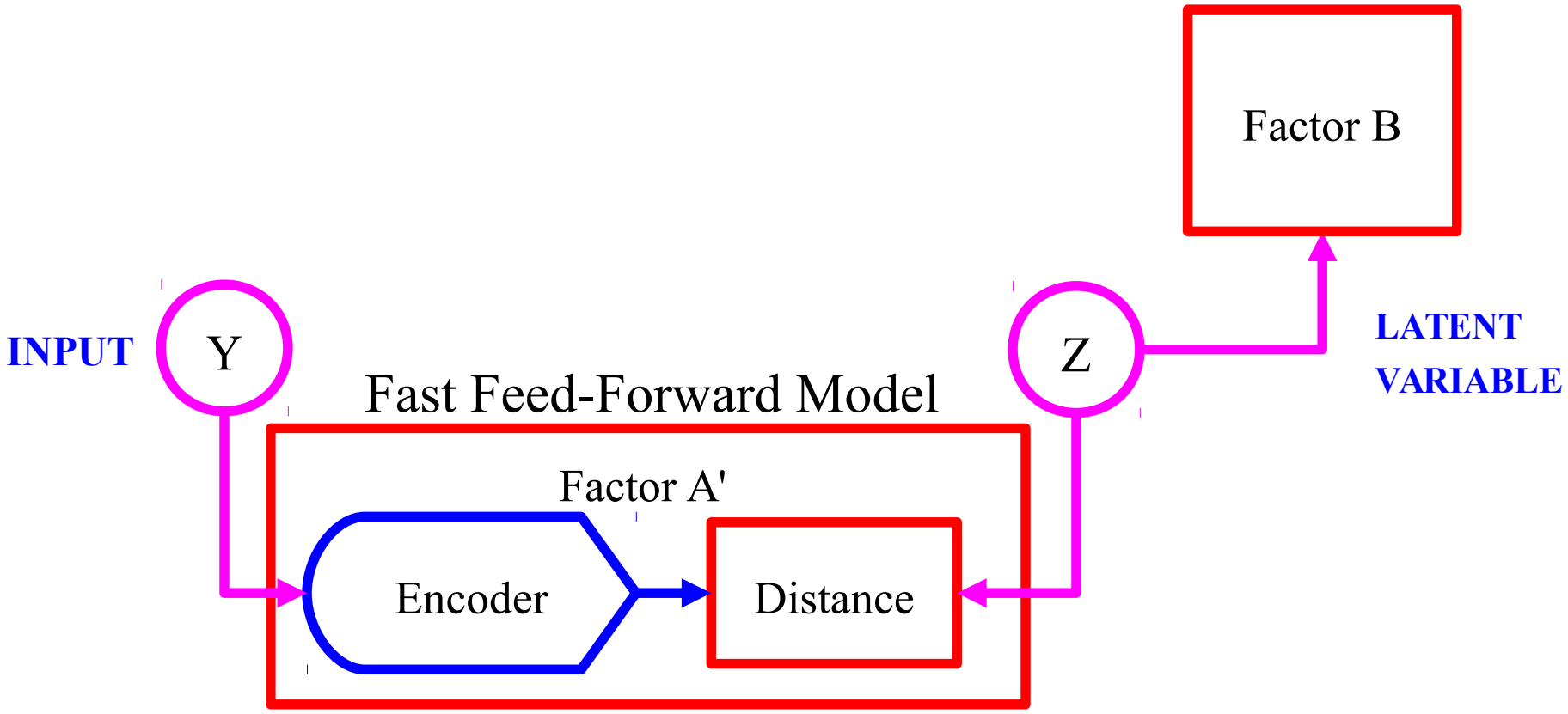
$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \lambda \sum_j |z_j|$$



- Inference is slow $Y \rightarrow \hat{Z} = argmin_Z E(Y, Z)$

Encoder Architecture

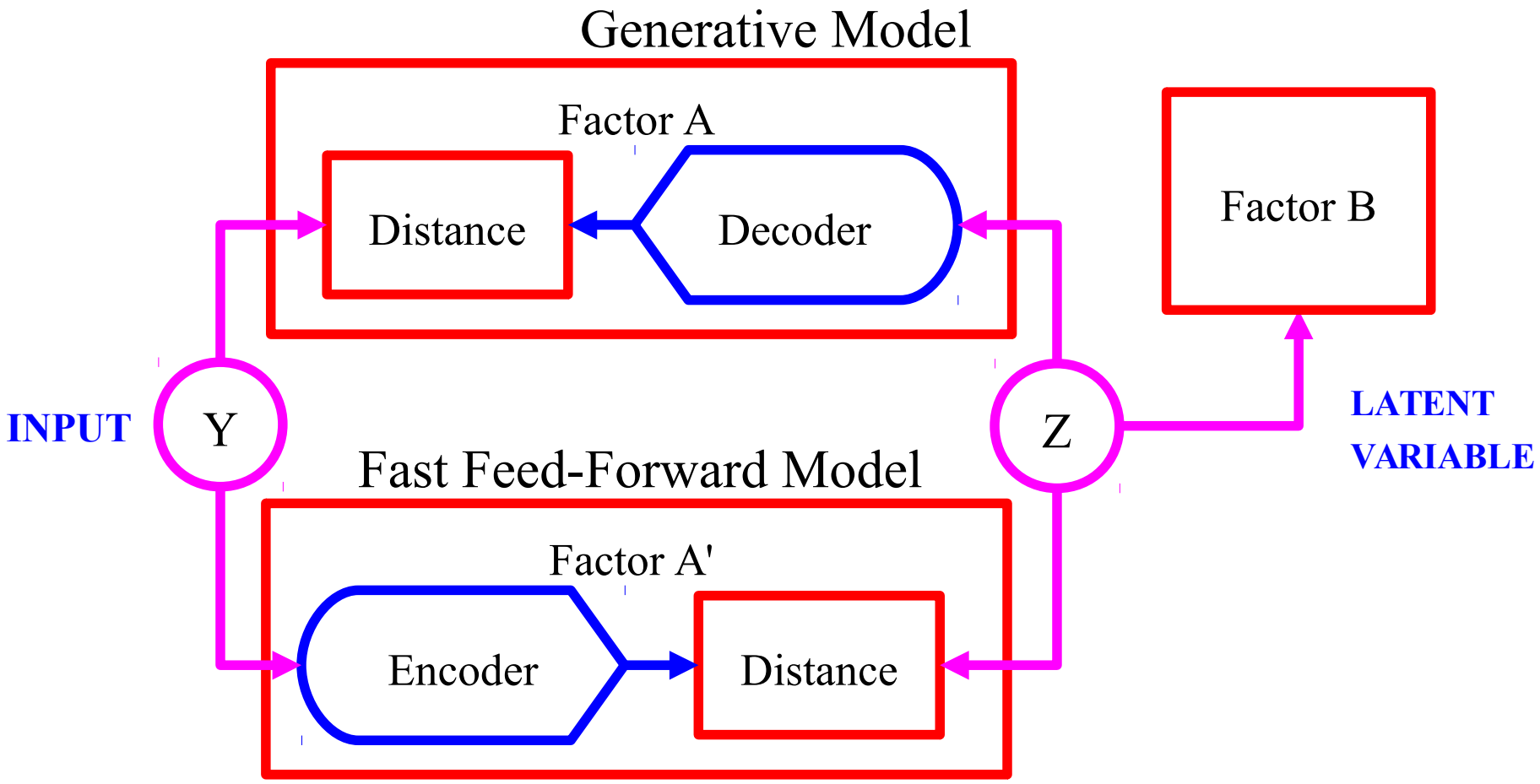
Examples: most ICA models, Product of Experts



Encoder-Decoder Architecture

[Kavukcuoglu, Ranzato, LeCun, rejected by every conference, 2008-2009]

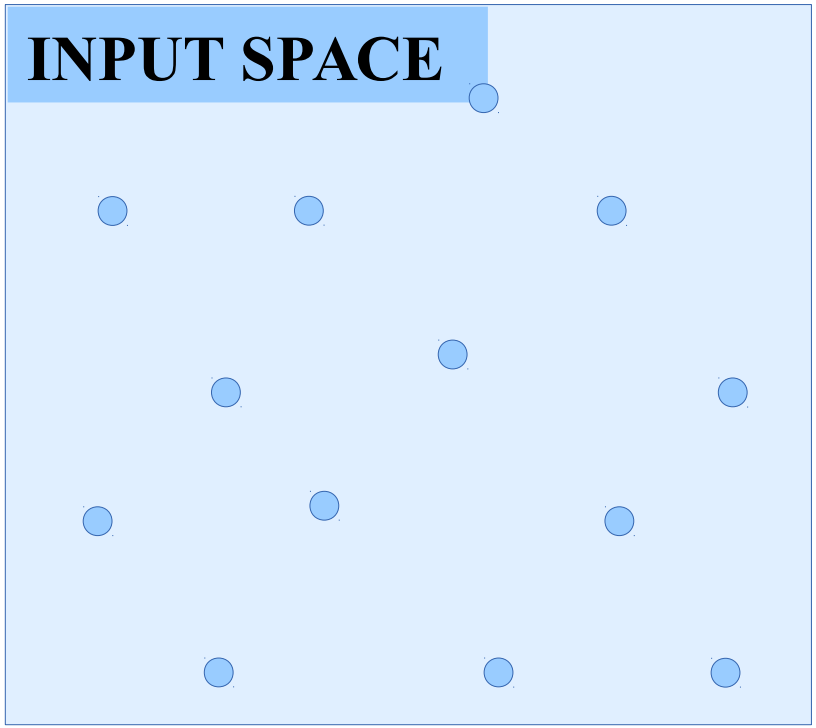
- Train a "simple" feed-forward function to predict the result of a complex optimization on the data points of interest

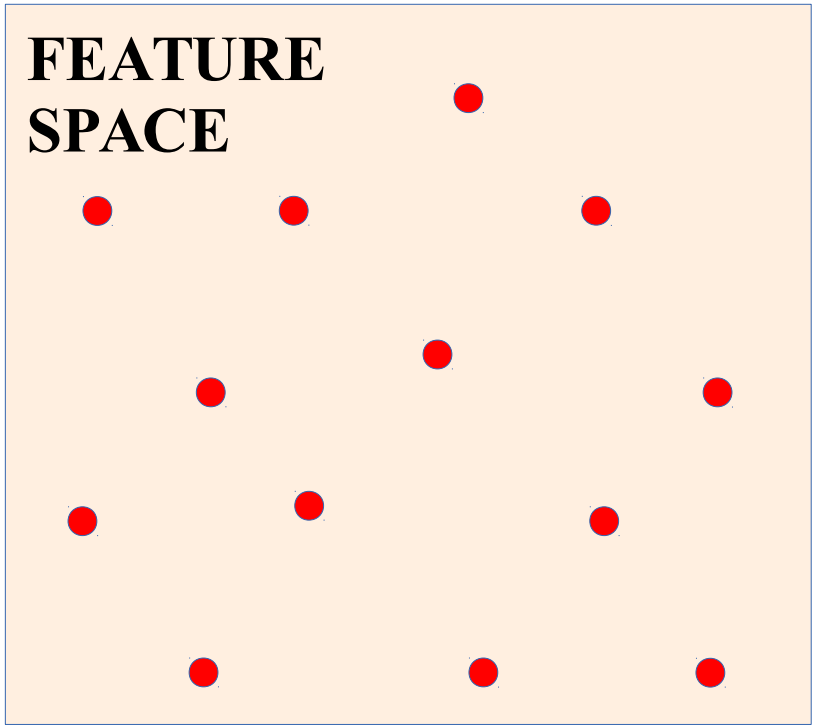


- 1. Find optimal Zi for all Yi; 2. Train Encoder to predict Zi from Yi

Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

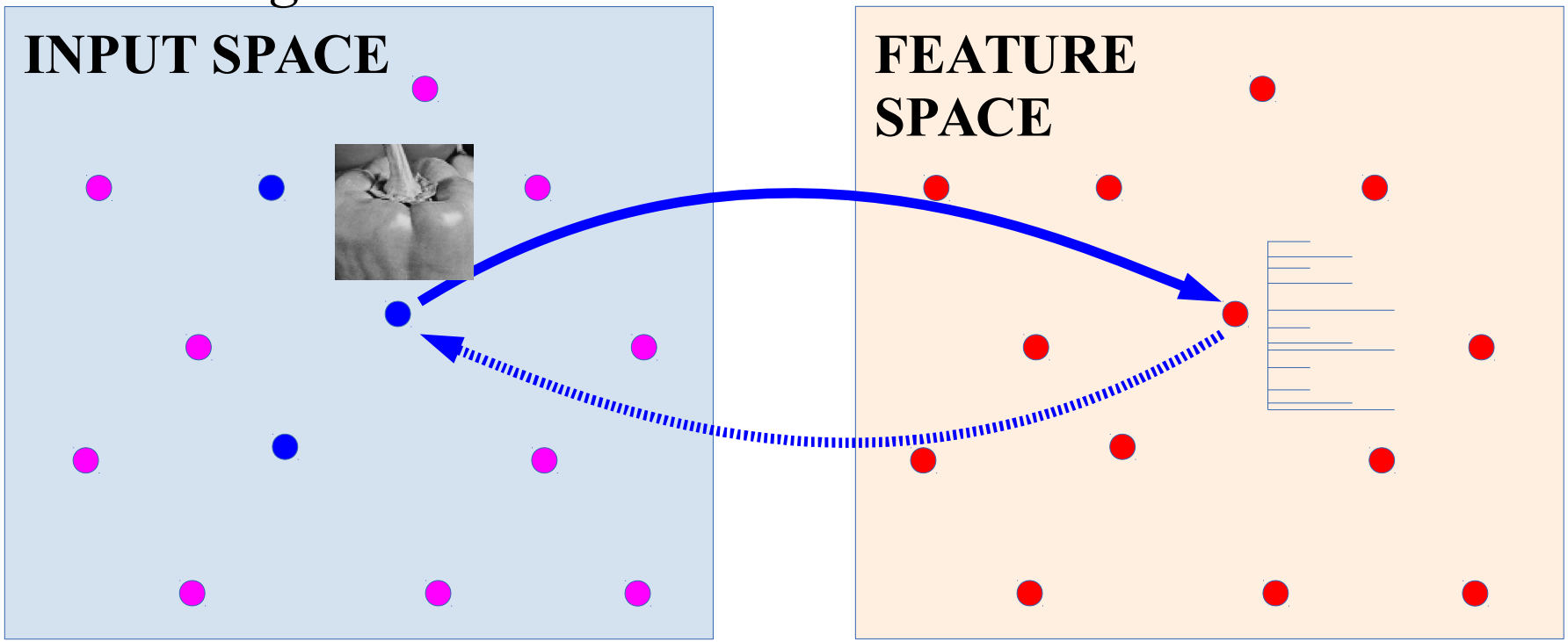




Why Limit the Information Content of the Code?

- **Training sample**
- **Input vector which is NOT a training sample**
- **Feature vector**

Training based on minimizing the reconstruction error over the training set

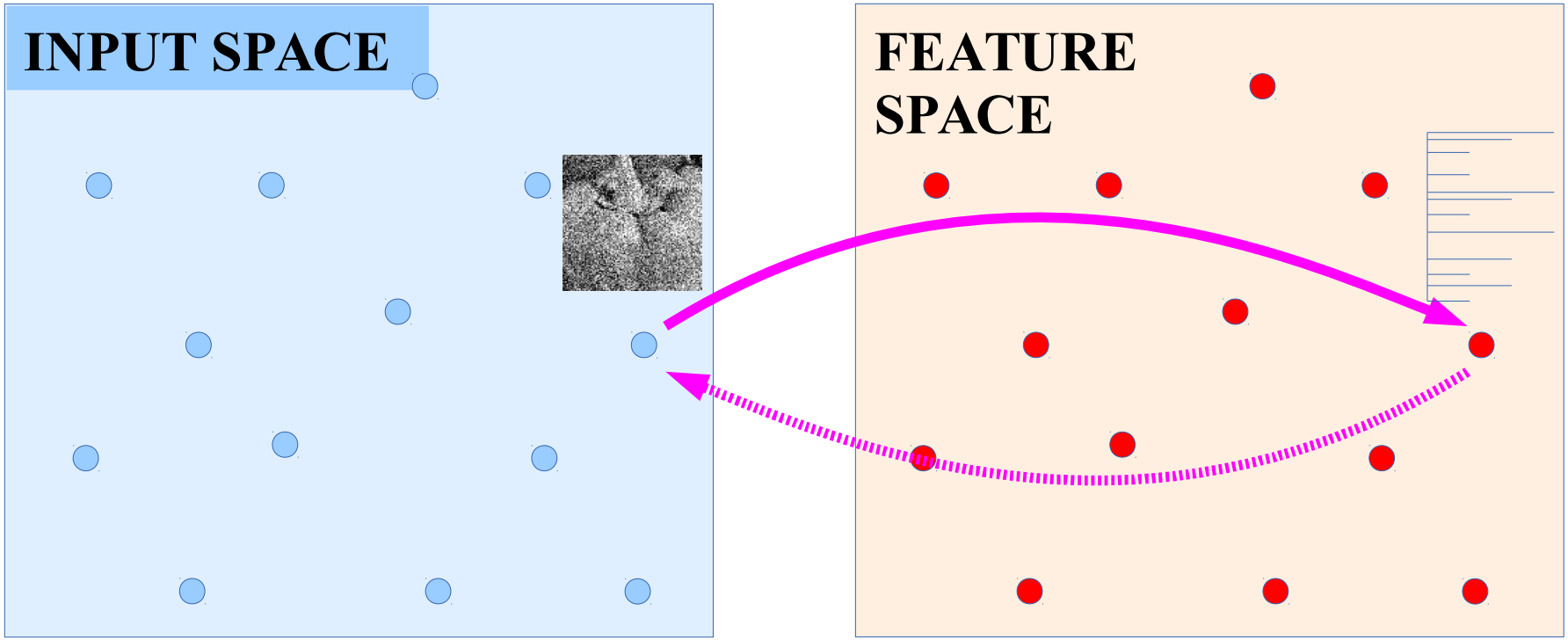


Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

BAD: machine does not learn structure from training data!!

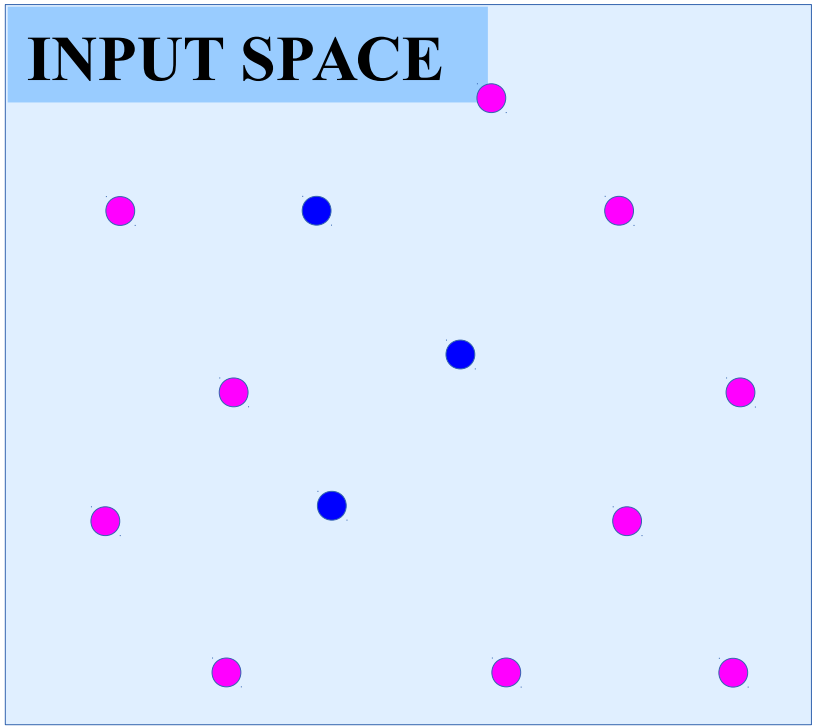
It just copies the data.

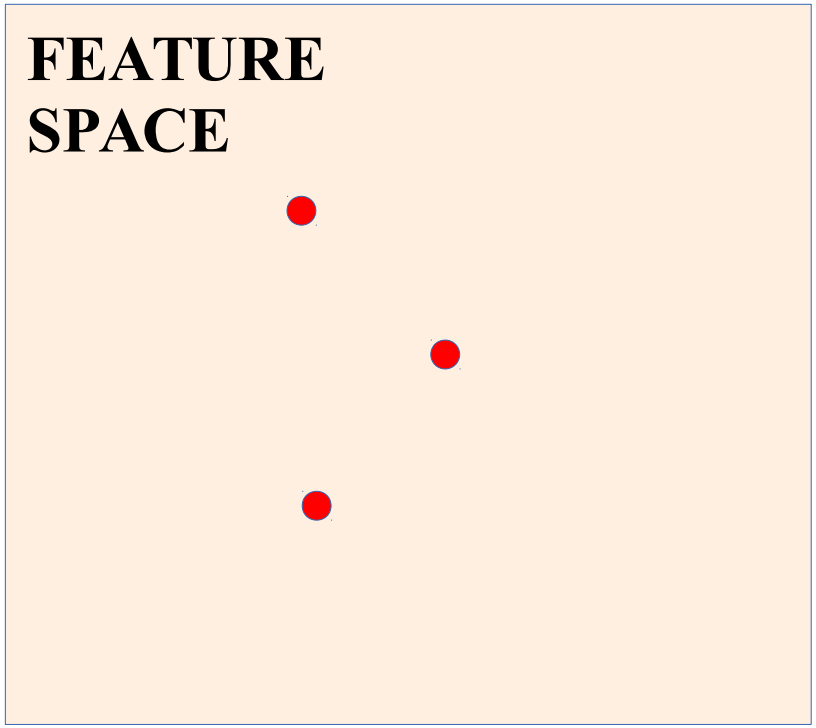


Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

IDEA: reduce number of available codes.

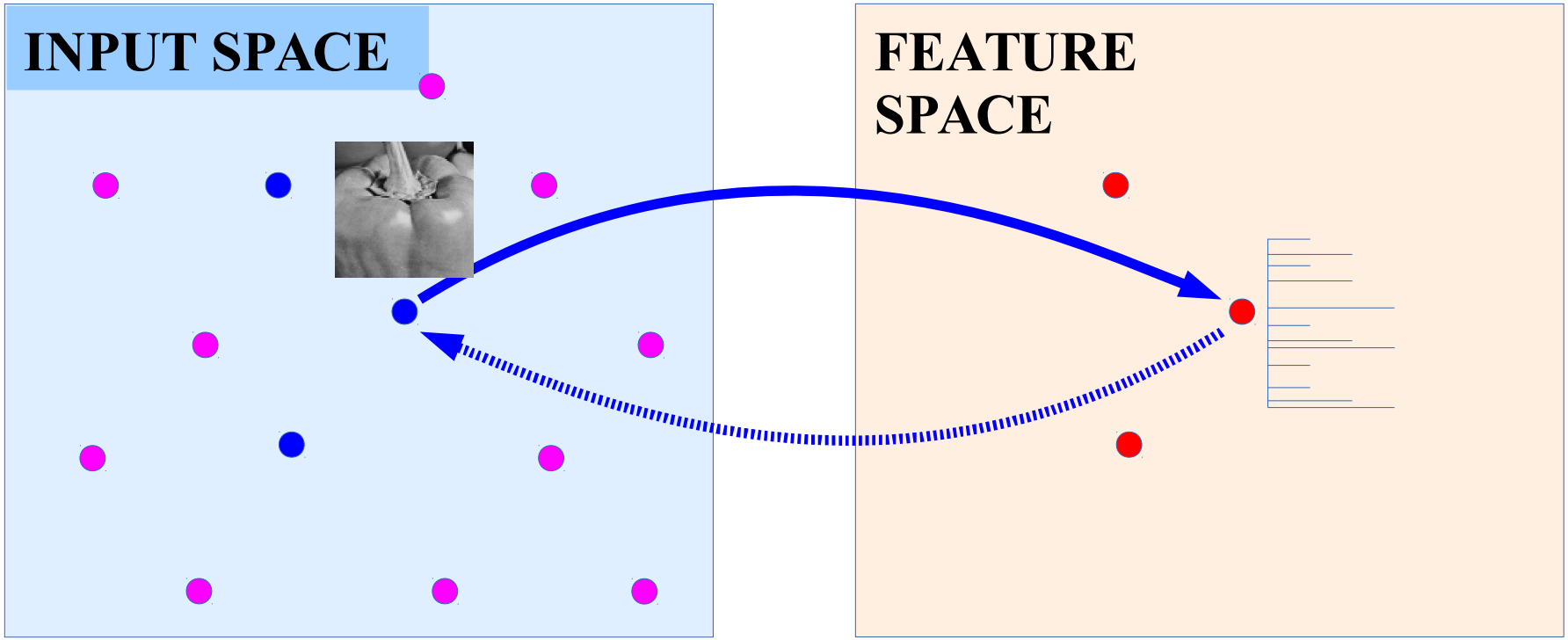




Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

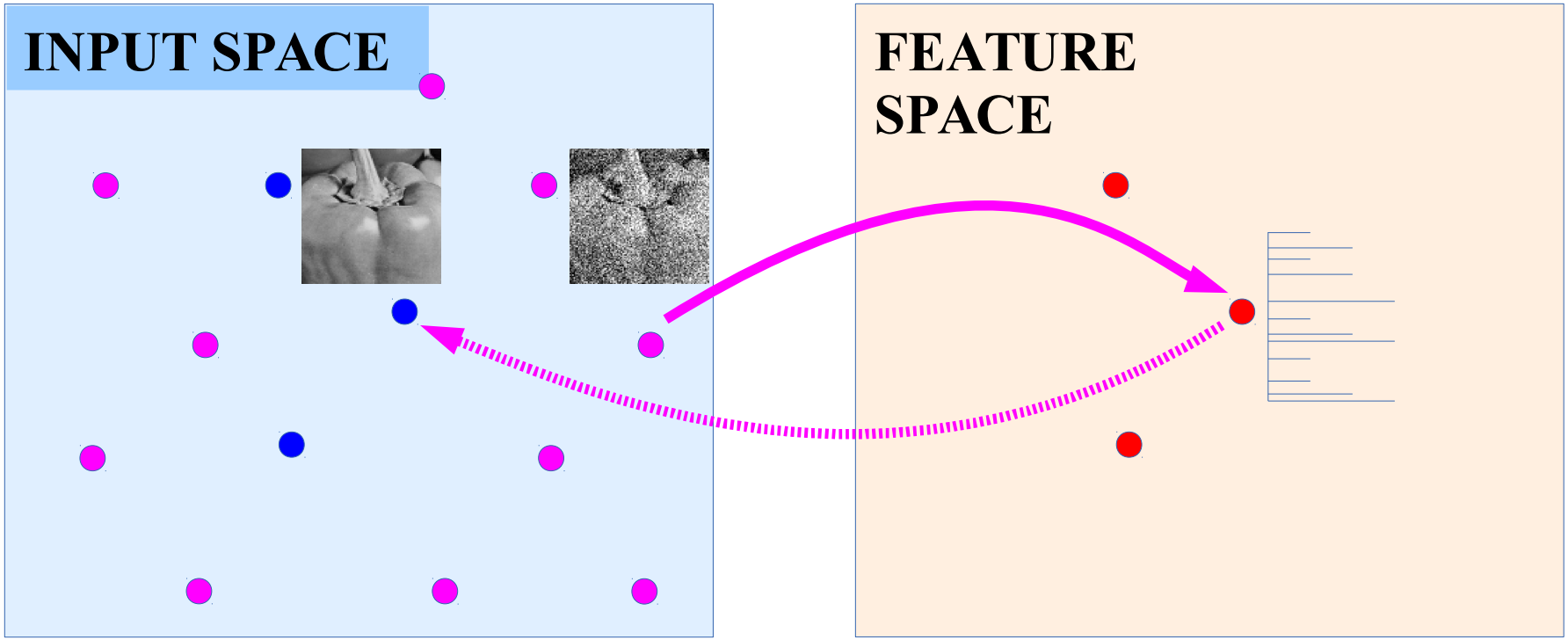
IDEA: reduce number of available codes.



Why Limit the Information Content of the Code?

- Training sample
- Input vector which is NOT a training sample
- Feature vector

IDEA: reduce number of available codes.



Predictive Sparse Decomposition (PSD): sparse auto-encoder

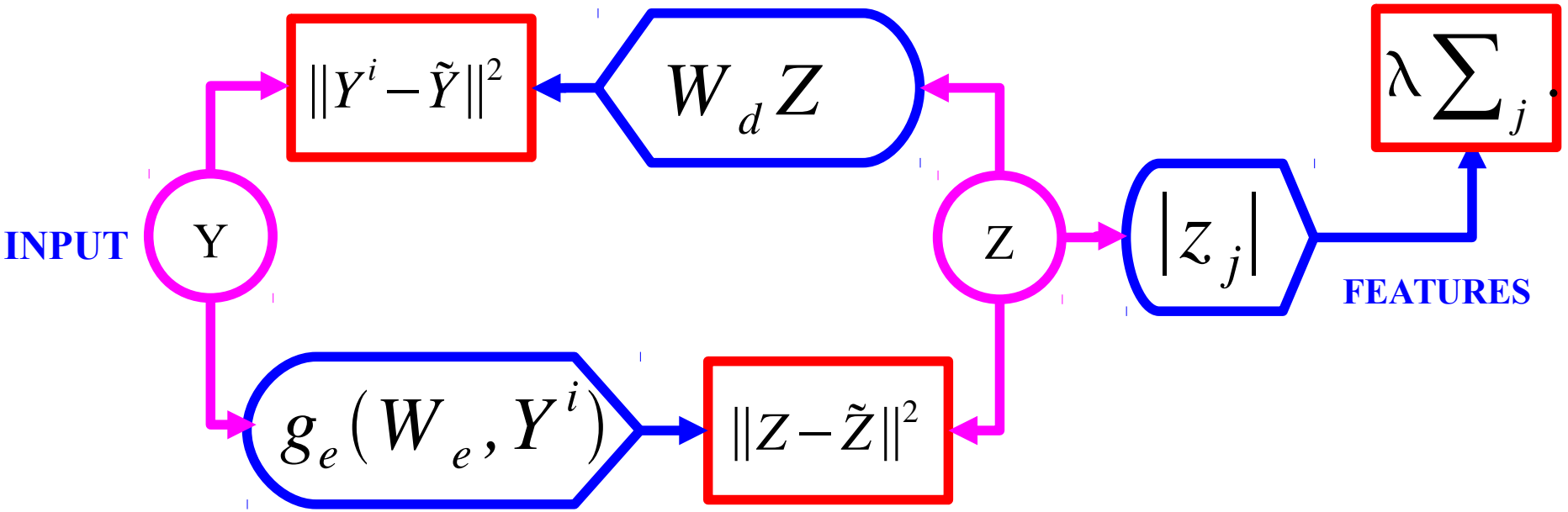
[Kavukcuoglu, Ranzato, LeCun, 2008 → arXiv:1010.3467],

Prediction the optimal code with a trained encoder

Energy = reconstruction_error + code_prediction_error + code_sparsity

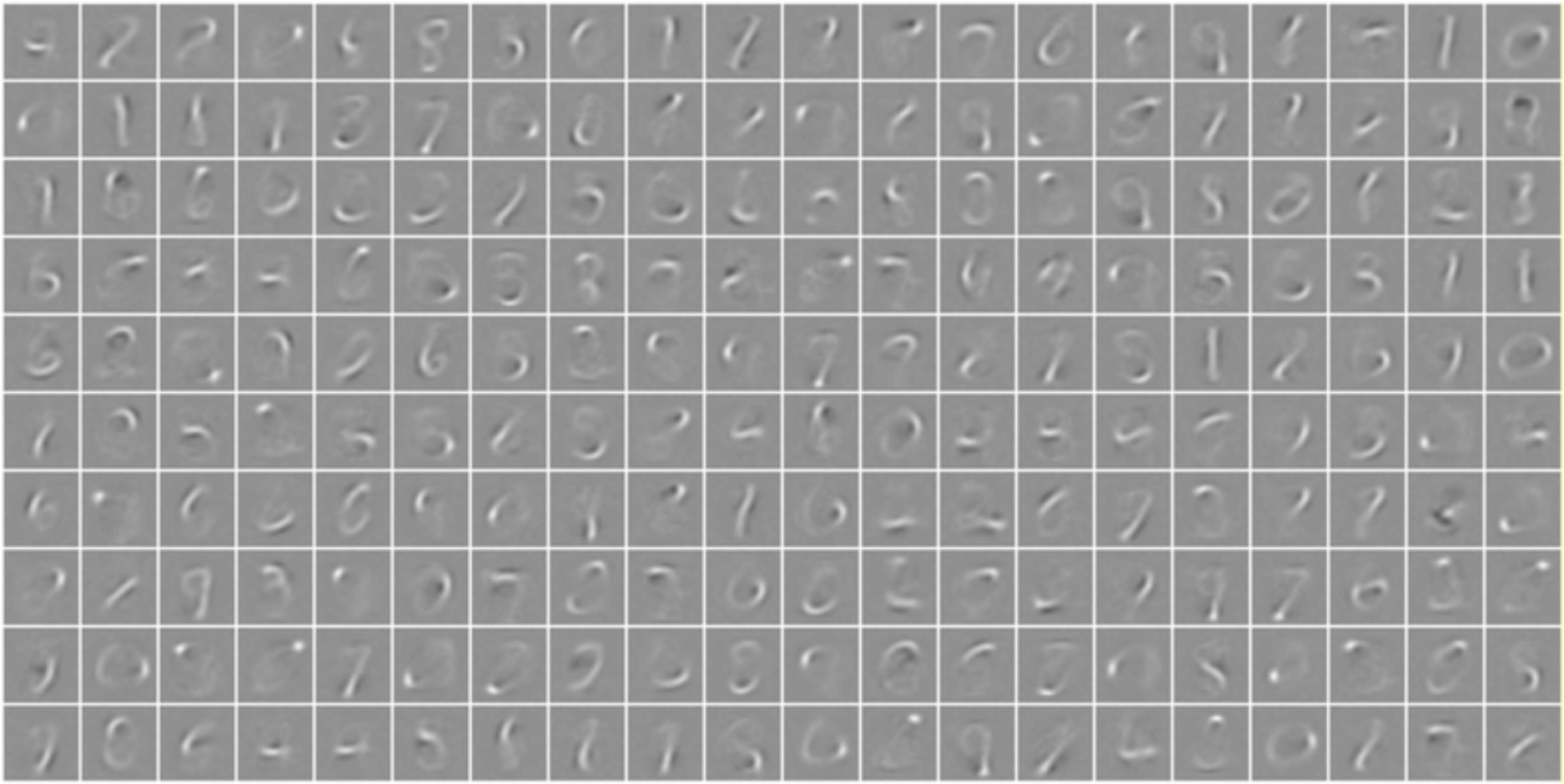
$$E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \|Z - g_e(W_e, Y^i)\|^2 + \lambda \sum_j |z_j|$$

$$g_e(W_e, Y^i) = shrinkage(W_e Y^i)$$



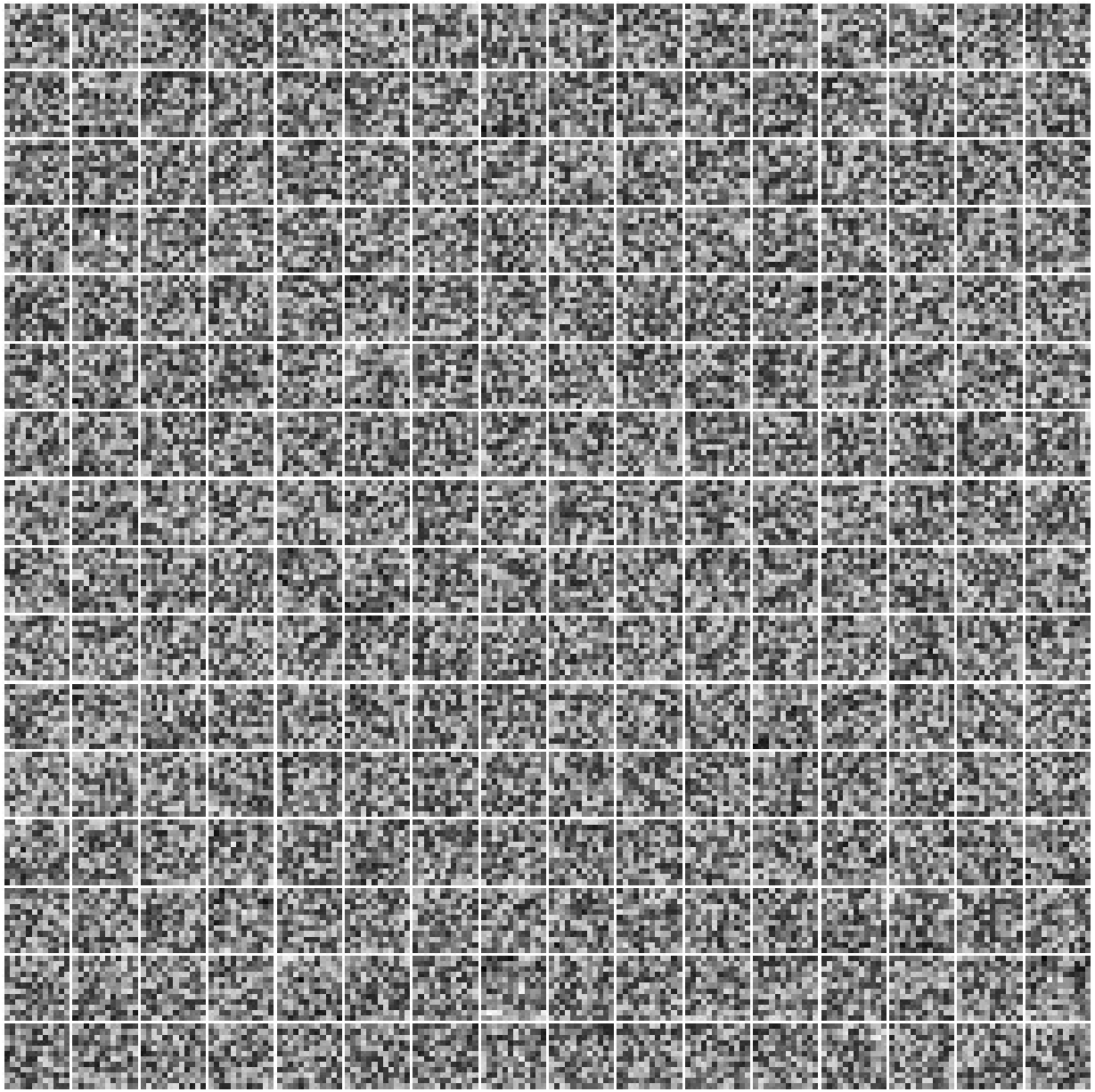
PSD: Basis Functions on MNIST

- Basis functions (and encoder matrix) are digit parts



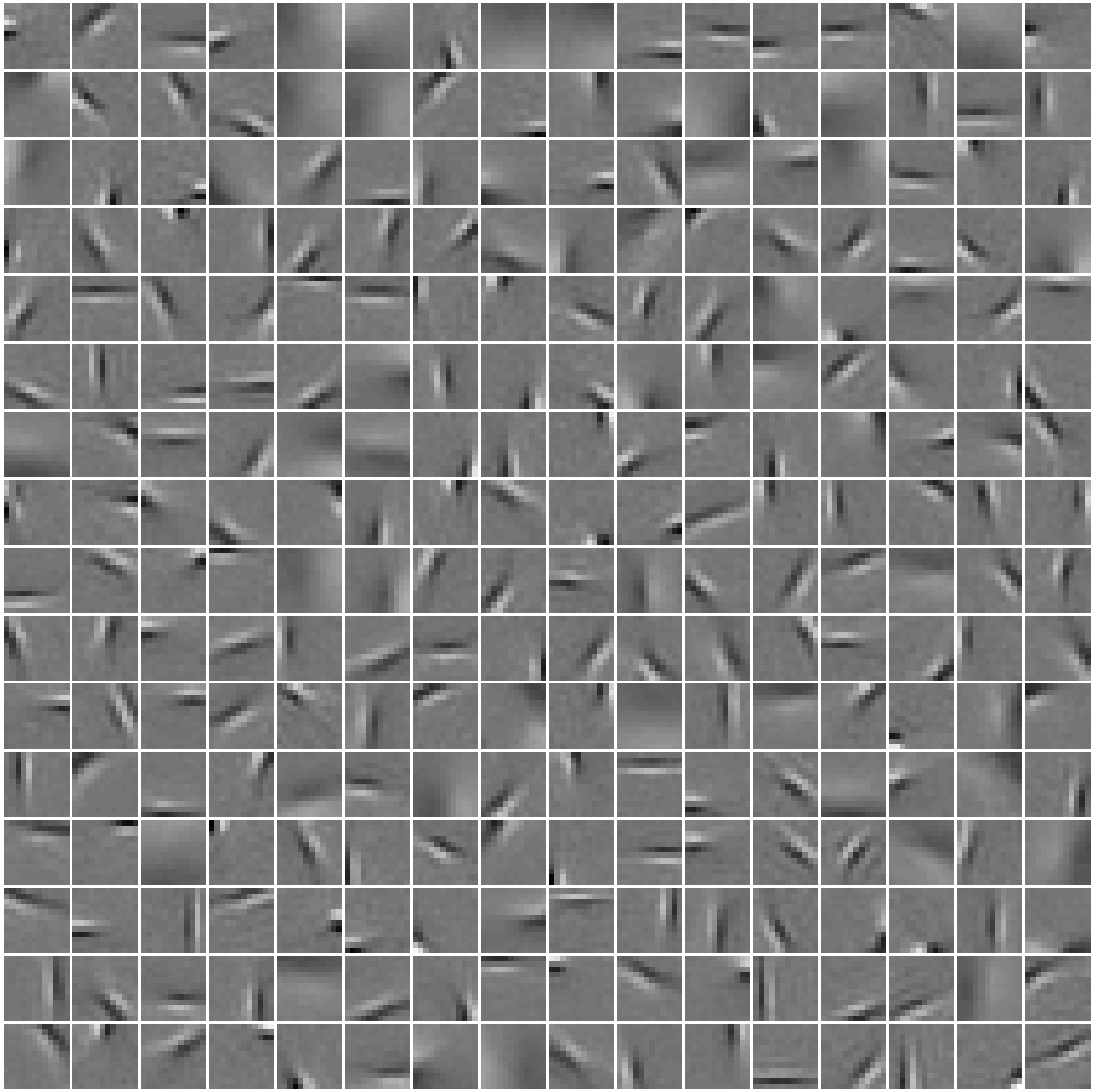
Predictive Sparse Decomposition (PSD): Training

- Training on natural images patches.
  - 12X12
  - 256 basis functions



iteration no 0

Learned Features on natural patches: V1-like receptive fields

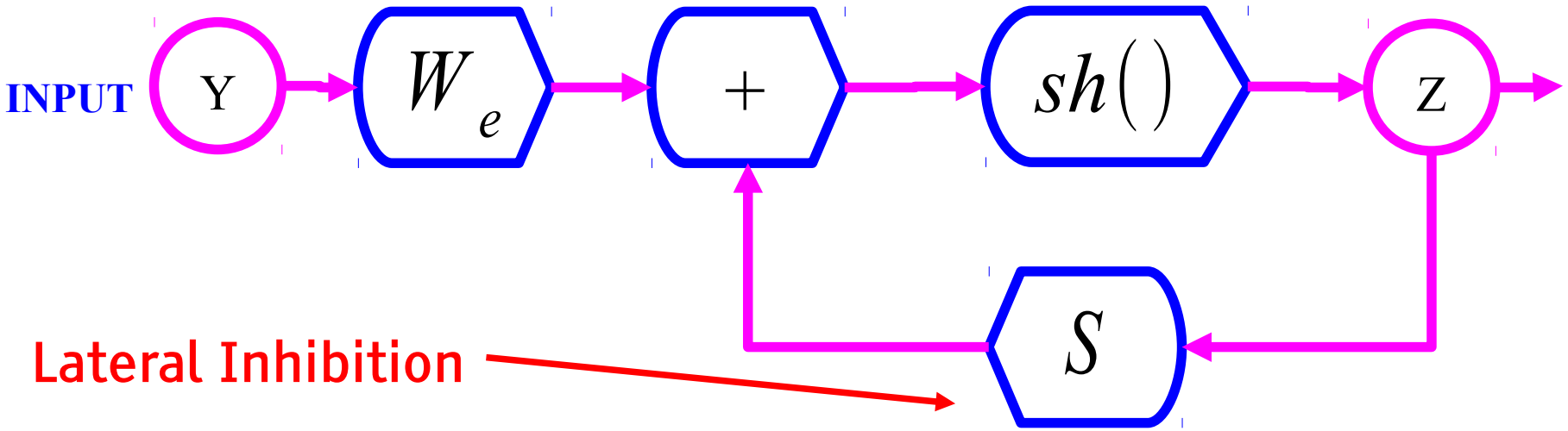



Learning to Infer
LISTA

Better Idea: Give the "right" structure to the encoder

- ISTA/FISTA: iterative algorithm that converges to optimal sparse code

[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

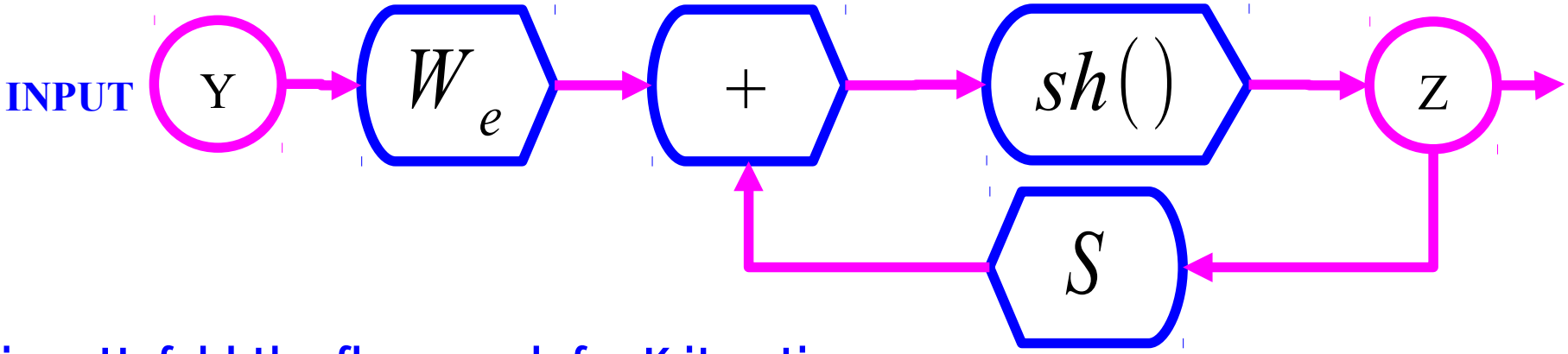


$$Z(t+1) = \text{Shrinkage}_{\lambda/L}\left[Z(t) - \frac{1}{L}W_d^T(W_d Z(t) - Y)\right]$$

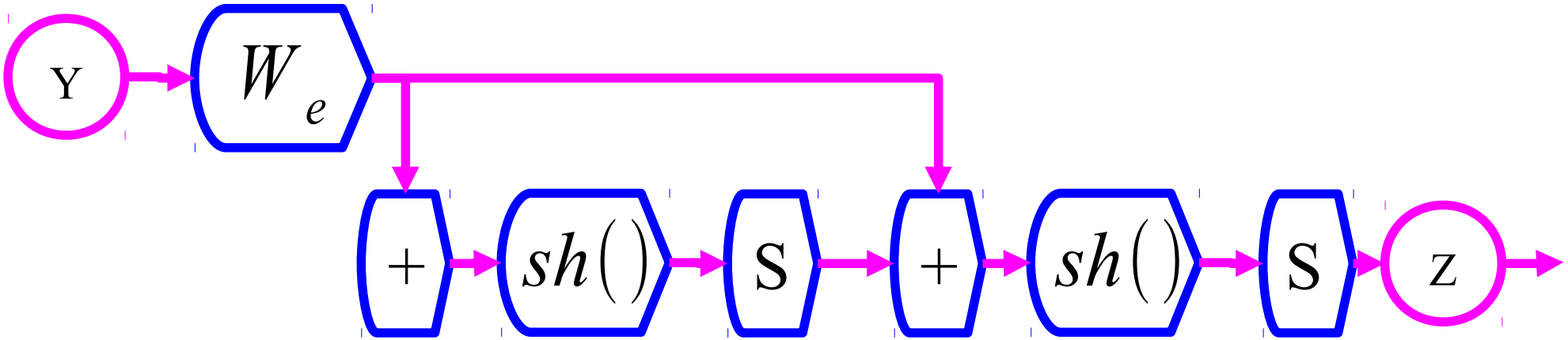
$$Z(t+1) = \text{Shrinkage}_{\lambda/L}\left[W_e^T Y + SZ(t)\right]; \quad W_e = \frac{1}{L}W_d; \quad S = I - \frac{1}{L}W_d^T W_d$$

LISTA: Train We and S matrices to give a good approximation quickly

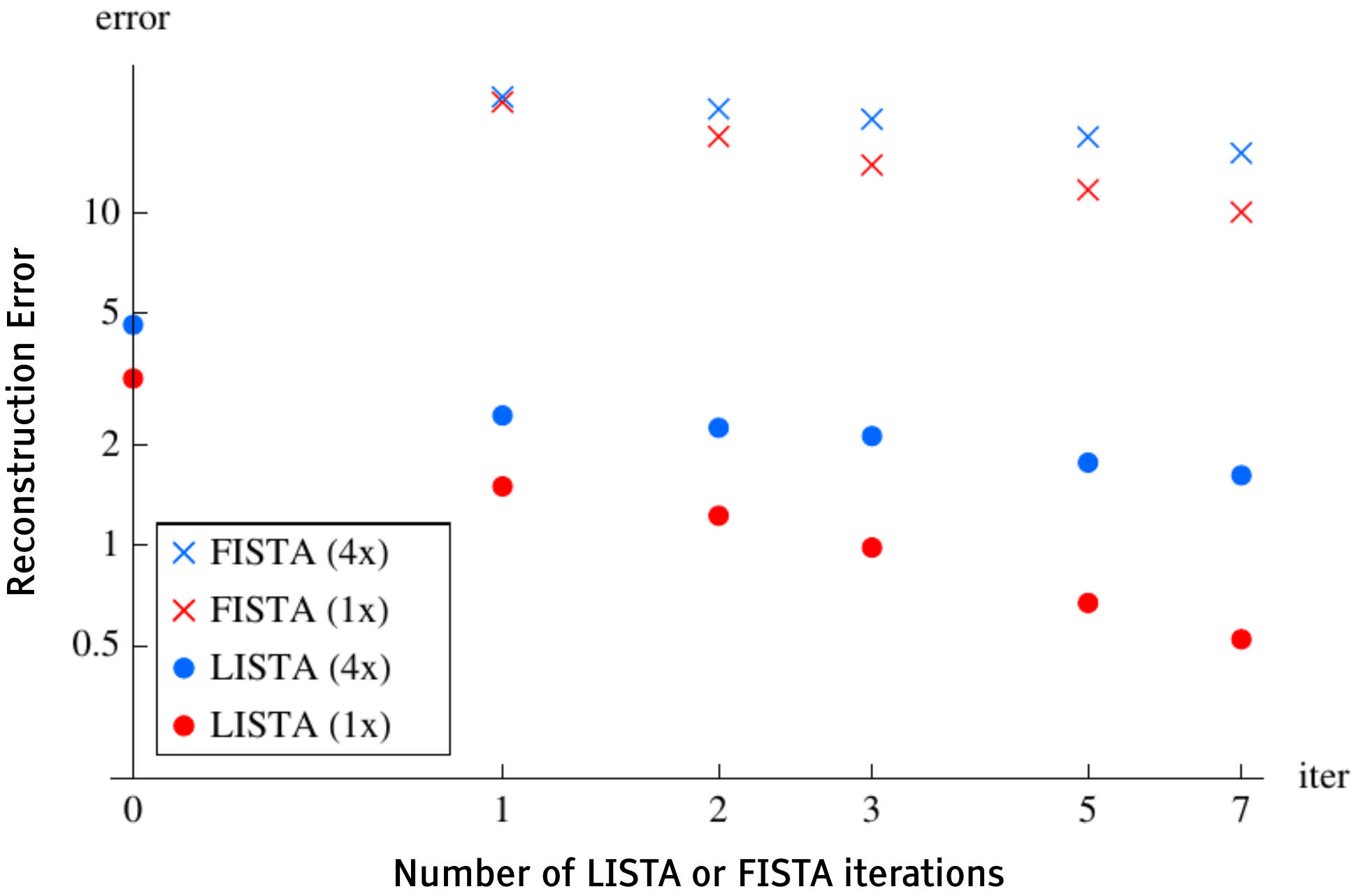
- Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters



- Time-Unfold the flow graph for K iterations
- Learn the We and S matrices with "backprop-through-time"
- Get the best approximate solution within K iterations

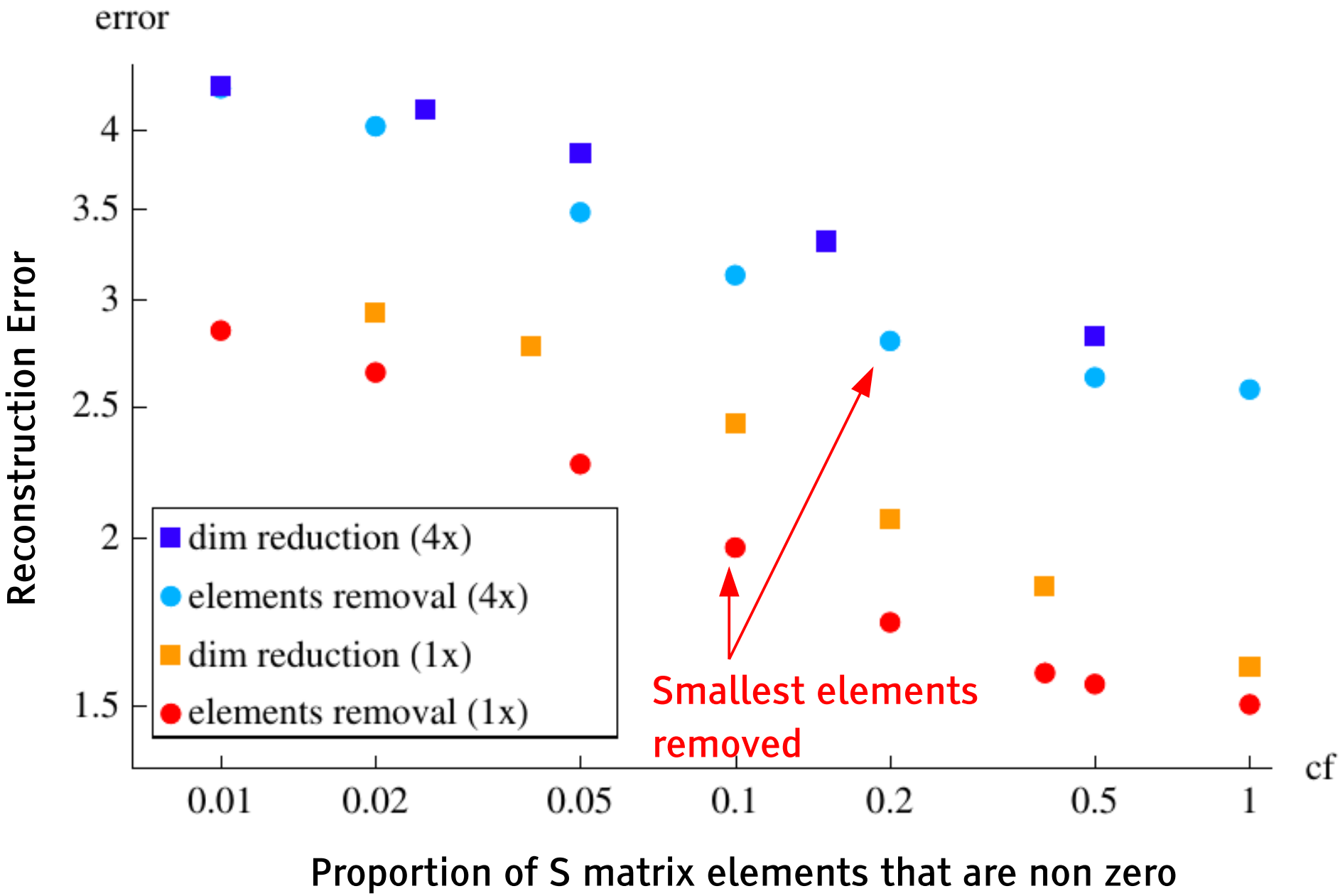


Learning ISTA (LISTA) vs ISTA/FISTA

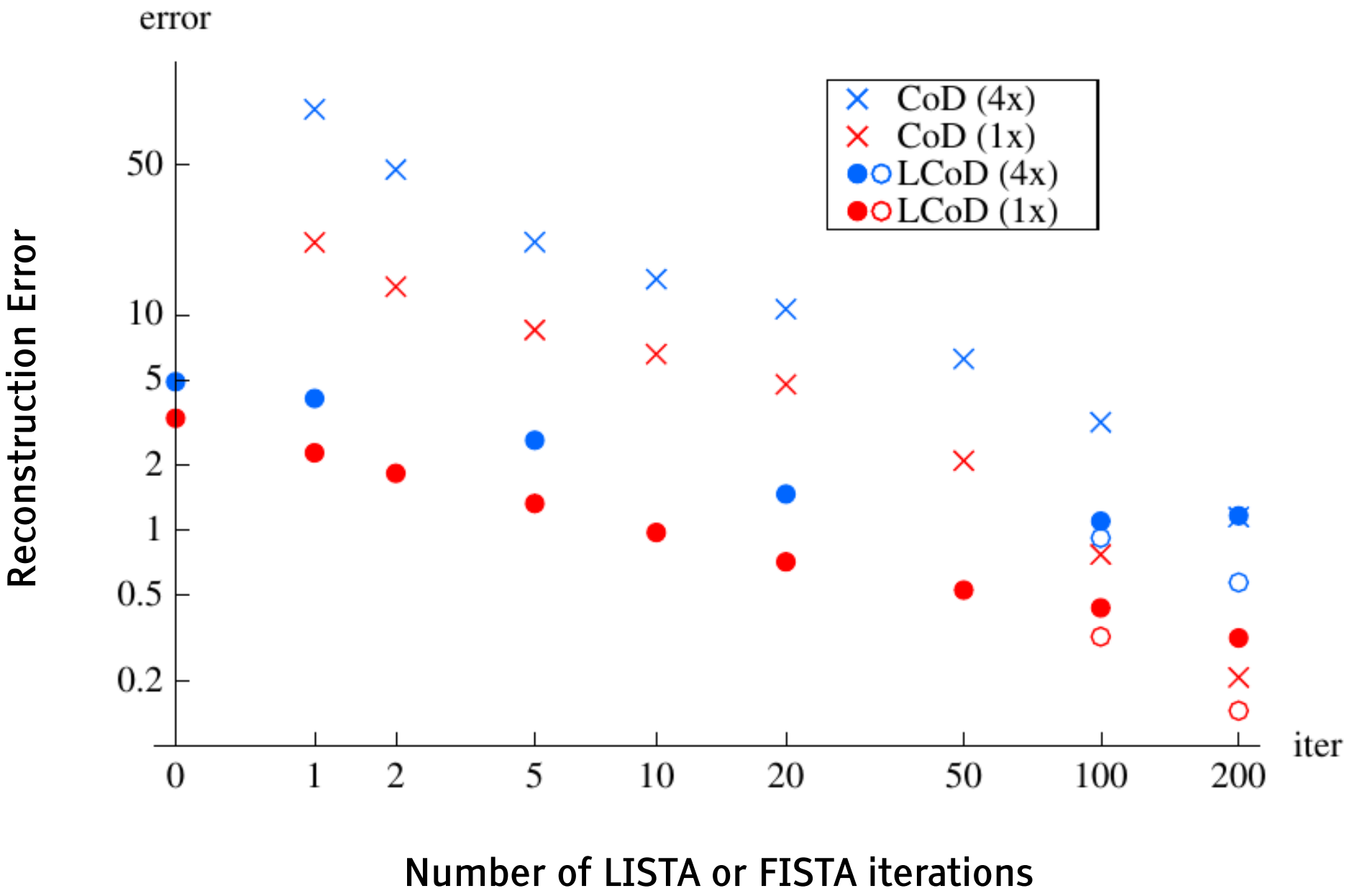


LISTA with partial mutual inhibition matrix

Y LeCun

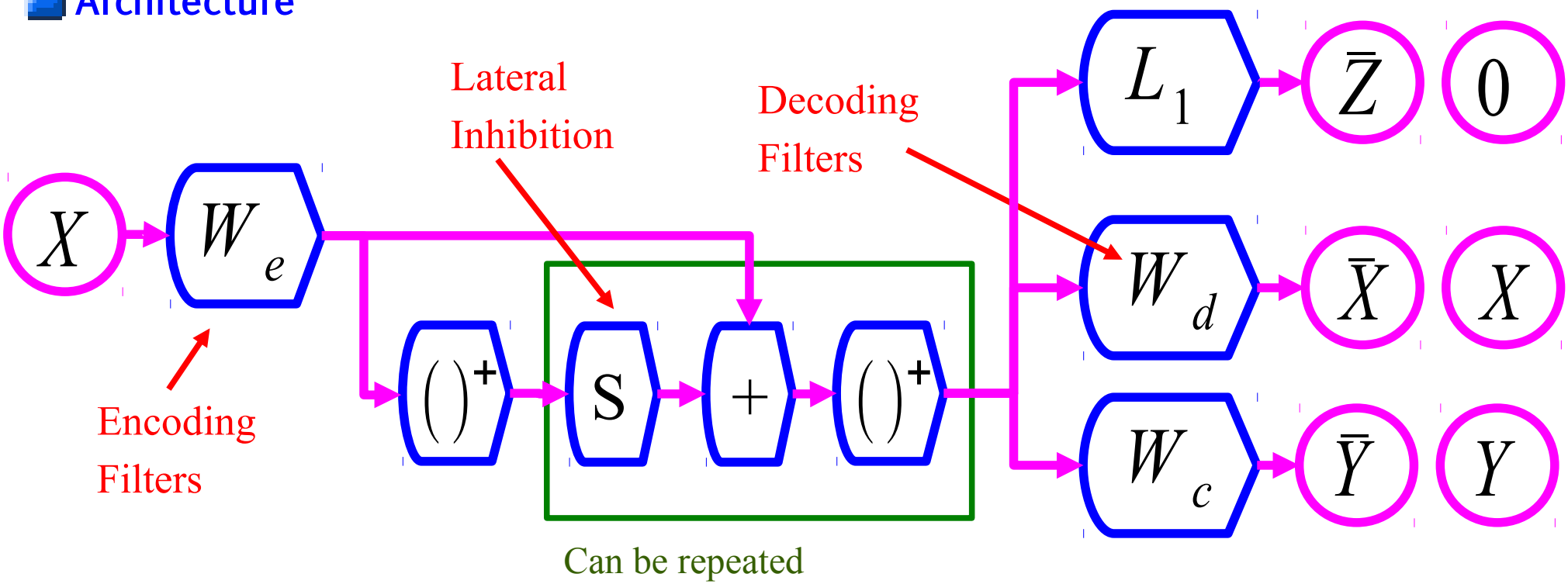


Learning Coordinate Descent (LcoD): faster than LISTA



Discriminative Recurrent Sparse Auto-Encoder (DrSAE)

- Architecture



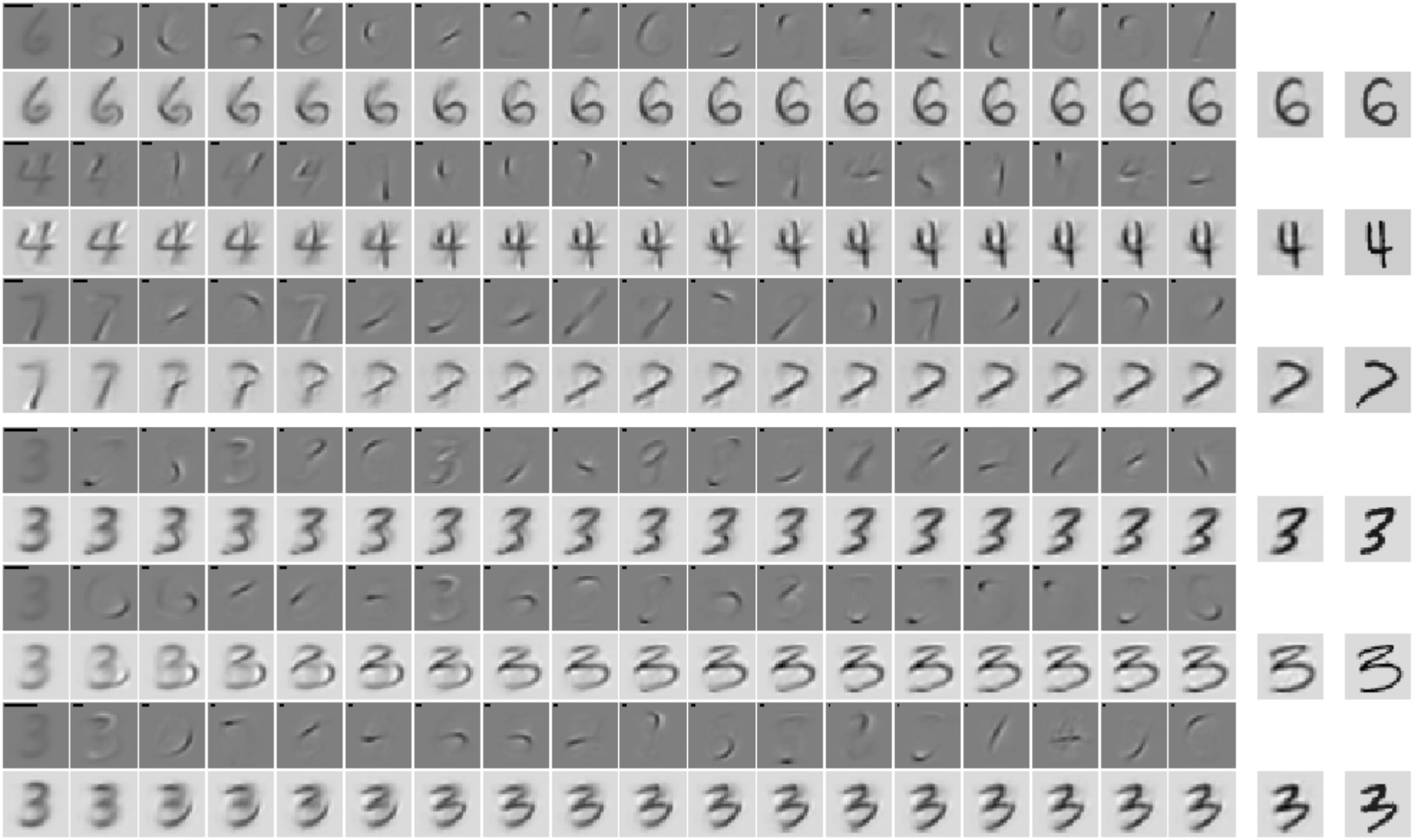
- Rectified linear units
- Classification loss: cross-entropy
- Reconstruction loss: squared error
- Sparsity penalty: L1 norm of last hidden layer
- Rows of Wd and columns of We constrained in unit sphere

[Rolfe & LeCun ICLR 2013]

DrSAE Discovers manifold structure of handwritten digits

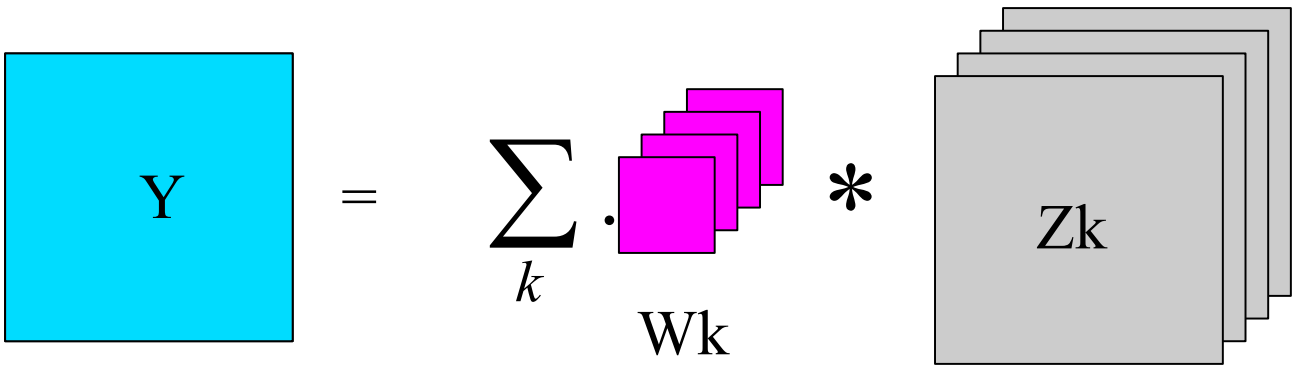
Y LeCun

- Image = prototype + sparse sum of "parts" (to move around the manifold)



Convolutional Sparse Coding

- **Replace the dot products with dictionary element by convolutions.**
  - Input Y is a full image
  - Each code component Zk is a feature map (an image)
  - Each dictionary element is a convolution kernel
- **Regular sparse coding** $E(Y,Z) = ||Y - \sum_k W_k Z_k||^2 + \alpha \sum_k |Z_k|$
- **Convolutional S.C.** $E(Y,Z) = ||Y - \sum_k W_k * Z_k||^2 + \alpha \sum_k |Z_k|$



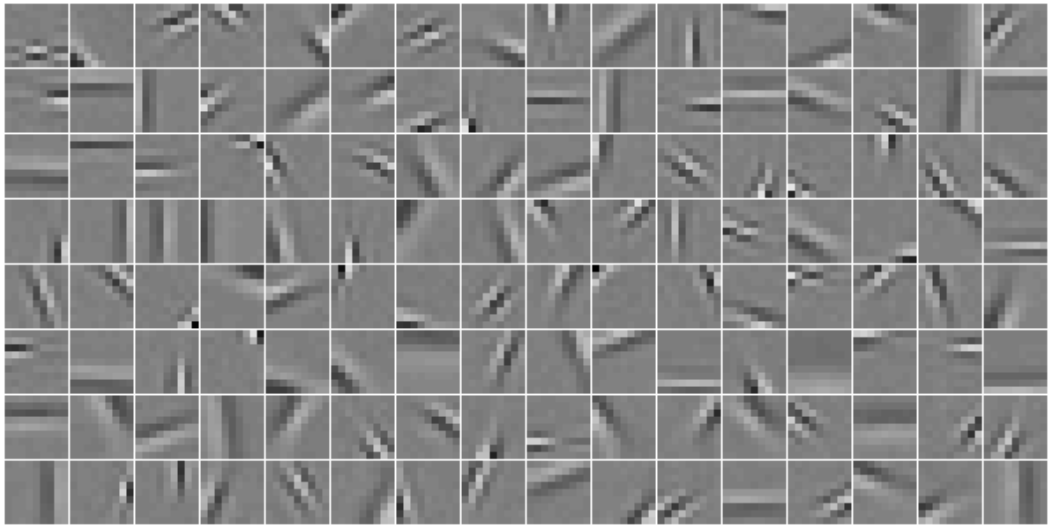
"deconvolutional networks" [Zeiler, Taylor, Fergus CVPR 2010]

Convolutional PSD: Encoder with a soft sh() Function

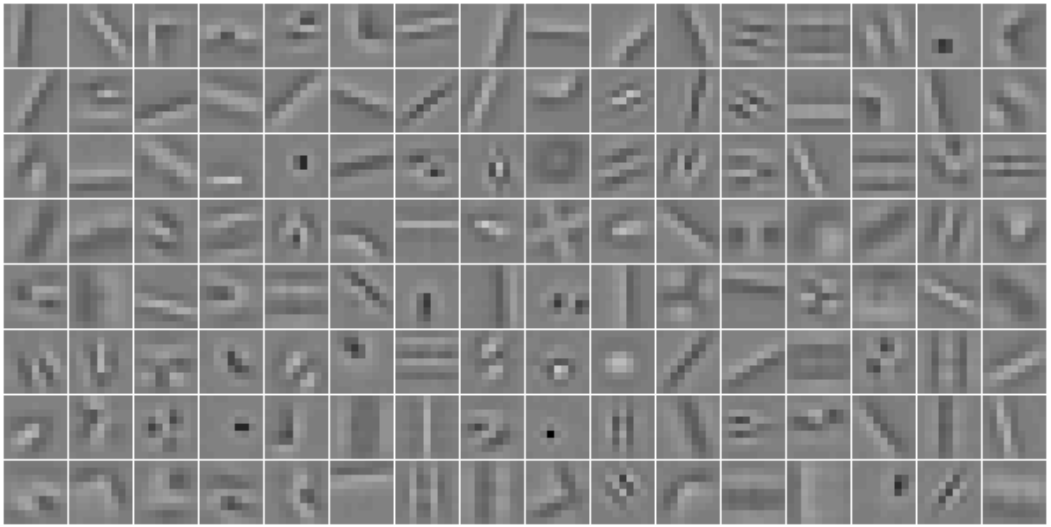
Convolutional Formulation

- Extend sparse coding from ***PATCH*** to ***IMAGE***

$$\mathcal{L}(x, z, \mathcal{D}) = \frac{1}{2}||x - \sum_{k=1}^{K} \mathcal{D}_k * z_k||_2^2 + \sum_{k=1}^{K} ||z_k - f(W^k * x)||_2^2 + |z|_1$$



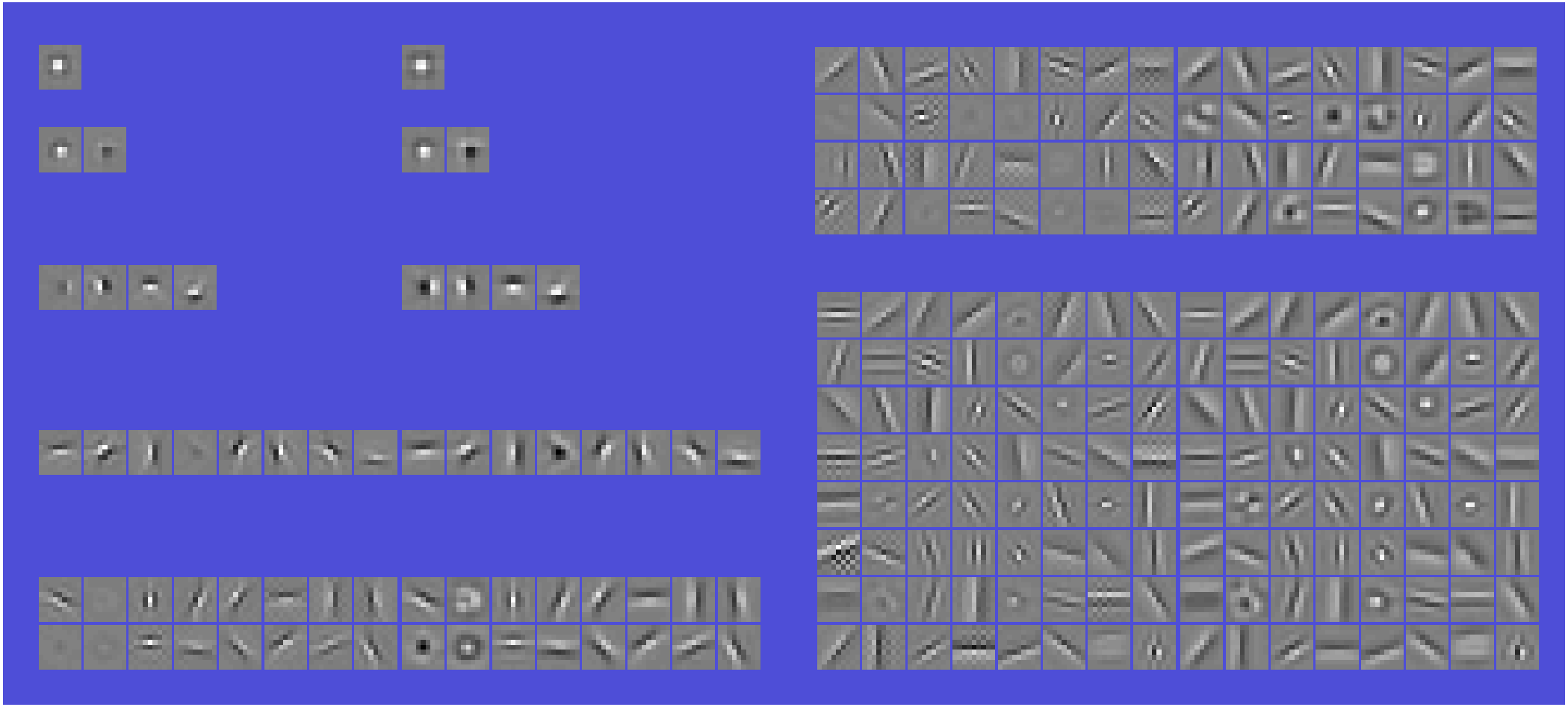
- ***PATCH*** based learning



- ***CONVOLUTIONAL*** learning

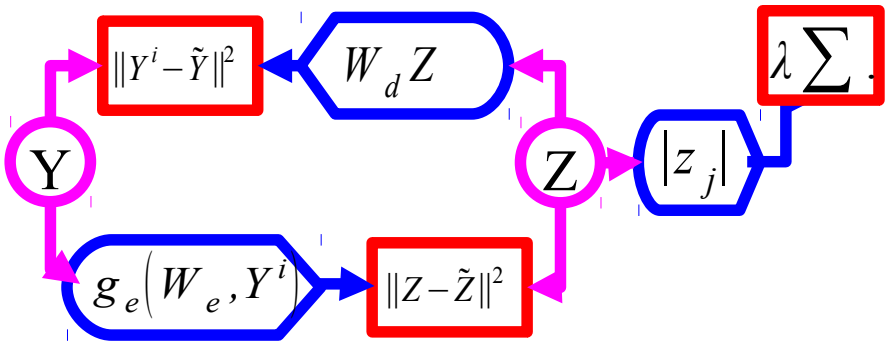
Convolutional Sparse Auto-Encoder on Natural Images

- Filters and Basis Functions obtained with 1, 2, 4, 8, 16, 32, and 64 filters.



Using PSD to Train a Hierarchy of Features

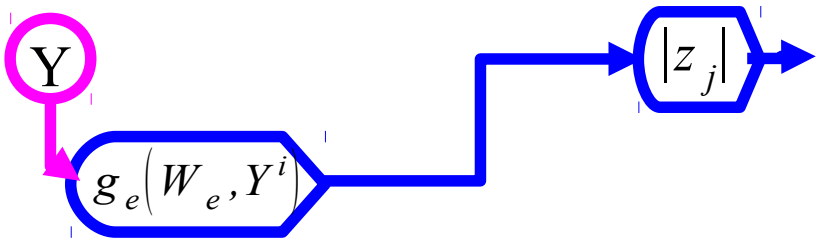
- Phase 1: train first layer using PSD



FEATURES

Using PSD to Train a Hierarchy of Features

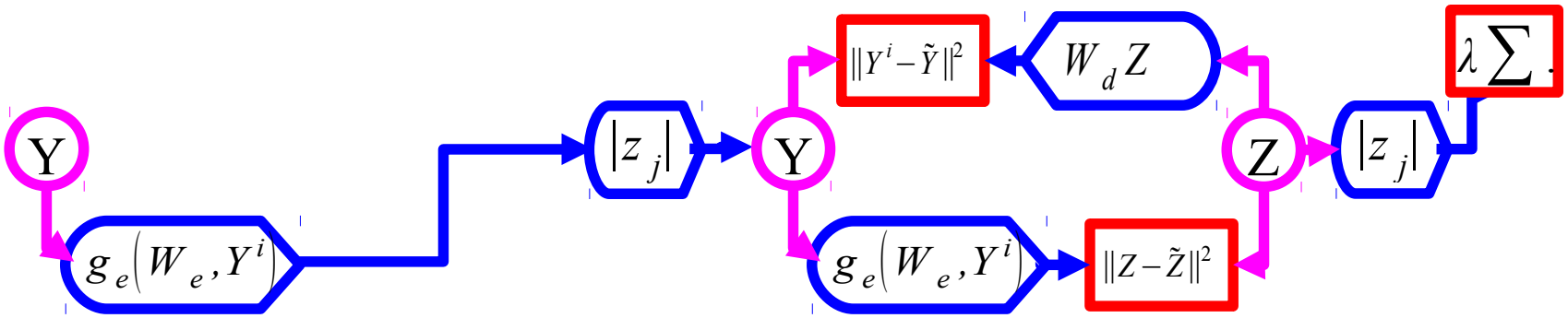
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor



FEATURES

Using PSD to Train a Hierarchy of Features

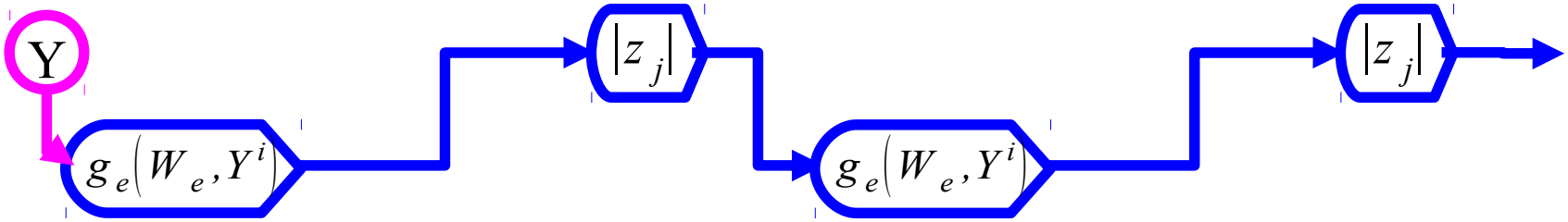
- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD



FEATURES

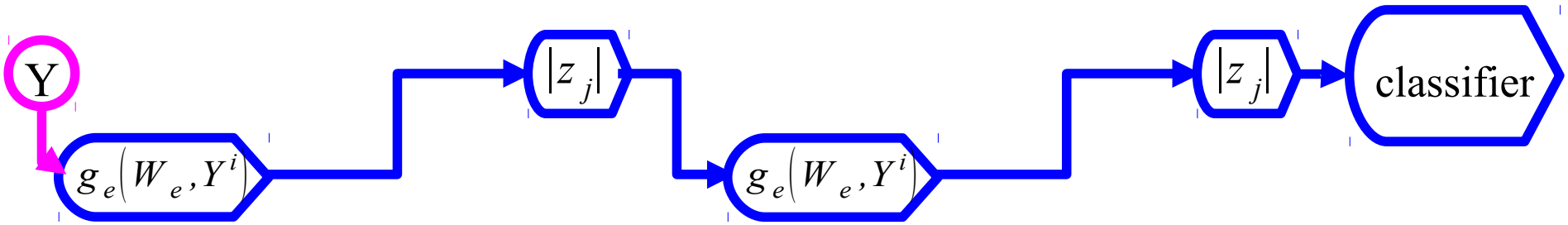
Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor

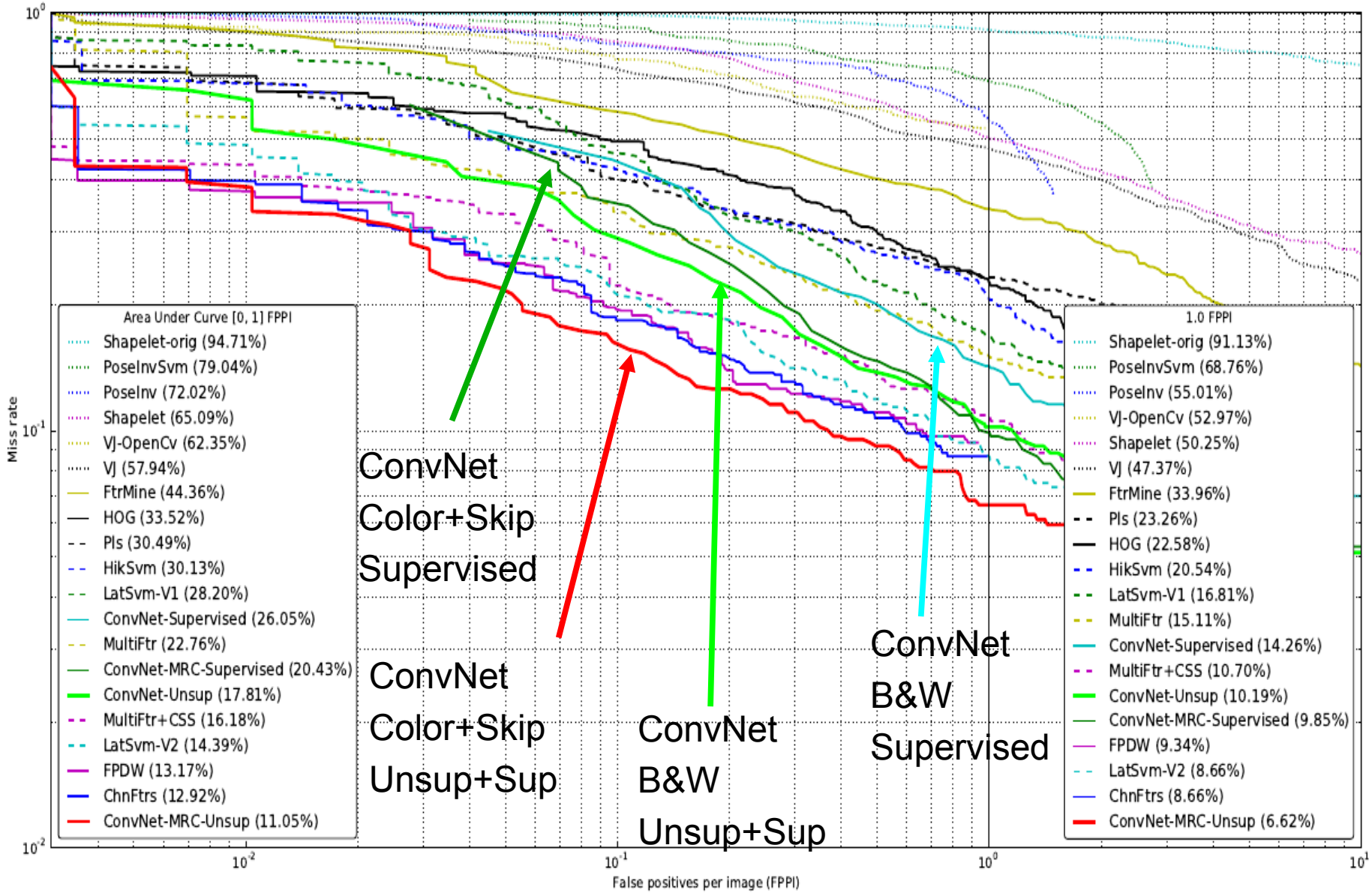


Using PSD to Train a Hierarchy of Features

- Phase 1: train first layer using PSD
- Phase 2: use encoder + absolute value as feature extractor
- Phase 3: train the second layer using PSD
- Phase 4: use encoder + absolute value as 2nd feature extractor
- Phase 5: train a supervised classifier on top
- Phase 6 (optional): train the entire system with supervised back-propagation



Pedestrian Detection: INRIA Dataset. Miss rate vs false positives



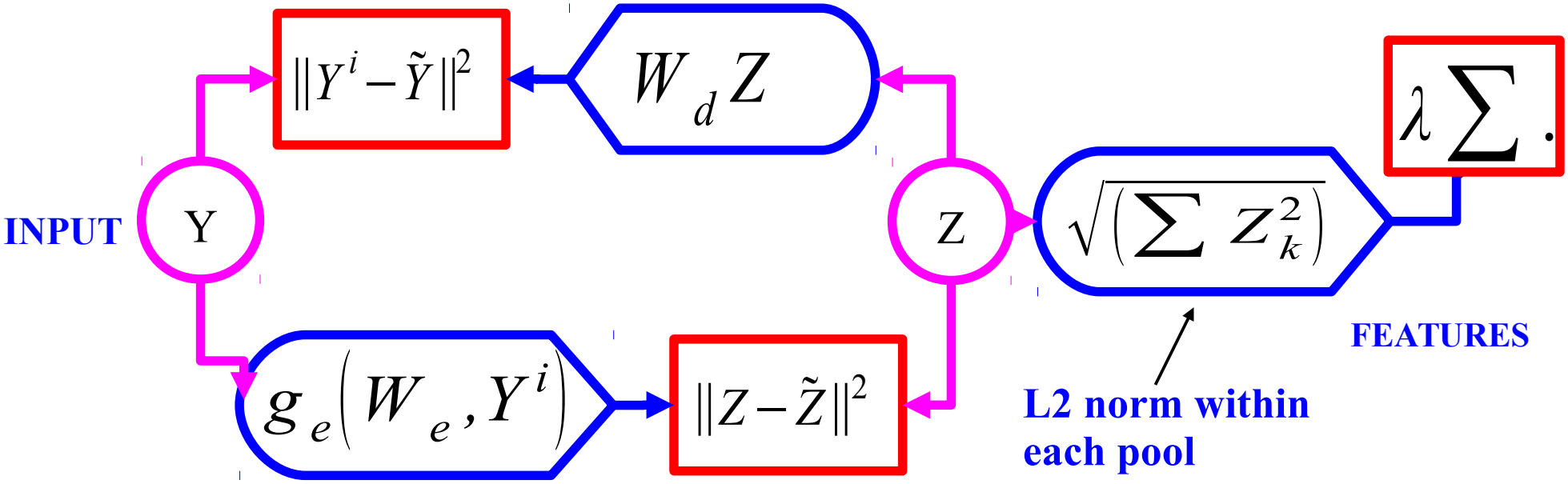
[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]


Unsupervised Learning:
Invariant Features

Learning Invariant Features with L2 Group Sparsity

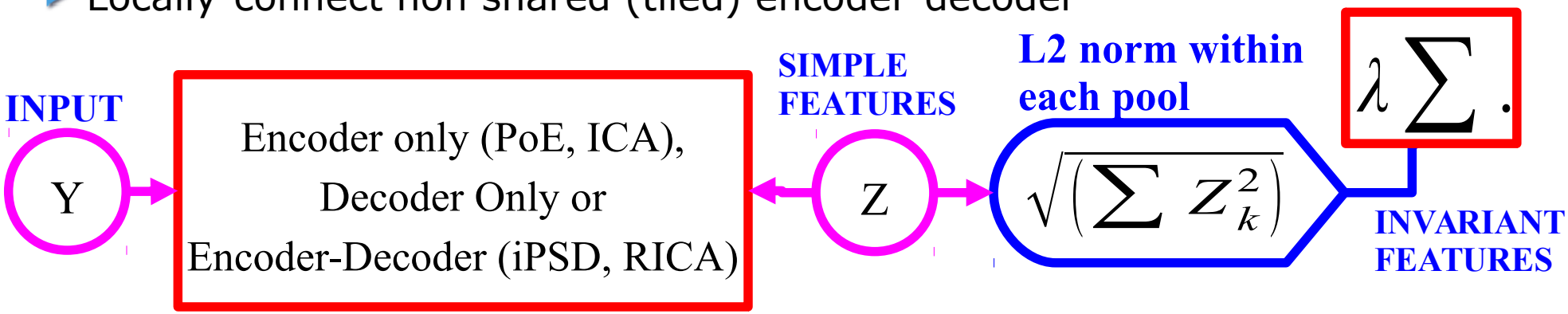
- Unsupervised PSD ignores the spatial pooling step.
- Could we devise a similar method that learns the pooling layer as well?
- Idea [Hyvarinen & Hoyer 2001]: group sparsity on pools of features
  - Minimum number of pools must be non-zero
  - Number of features that are on within a pool doesn't matter
  - Pools tend to regroup similar features

$$E(Y,Z) = \|Y - W_d Z\|^2 + \|Z - g_e(W_e, Y)\|^2 + \sum_j \sqrt{\sum_{k \in P_j} Z_k^2}$$



Learning Invariant Features with L2 Group Sparsity

- Idea: features are pooled in group.
  - Sparsity: sum over groups of L2 norm of activity in group.
- [Hyvärinen Hoyer 2001]: "subspace ICA"
  - decoder only, square
- [Welling, Hinton, Osindero NIPS 2002]: pooled product of experts
  - encoder only, overcomplete, log student-T penalty on L2 pooling
- [Kavukcuoglu, Ranzato, Fergus LeCun, CVPR 2010]: Invariant PSD
  - encoder-decoder (like PSD), overcomplete, L2 pooling
- [Le et al. NIPS 2011]: Reconstruction ICA
  - Same as [Kavukcuoglu 2010] with linear encoder and tied decoder
- [Gregor & LeCun arXiv:1006:0448, 2010] [Le et al. ICML 2012]
  - Locally-connect non shared (tiled) encoder-decoder



Groups are local in a 2D Topographic Map

- The filters arrange themselves spontaneously so that similar filters enter the same pool.
- The pooling units can be seen as complex cells
- Outputs of pooling units are invariant to local transformations of the input
  - For some it's translations, for others rotations, or other transformations.

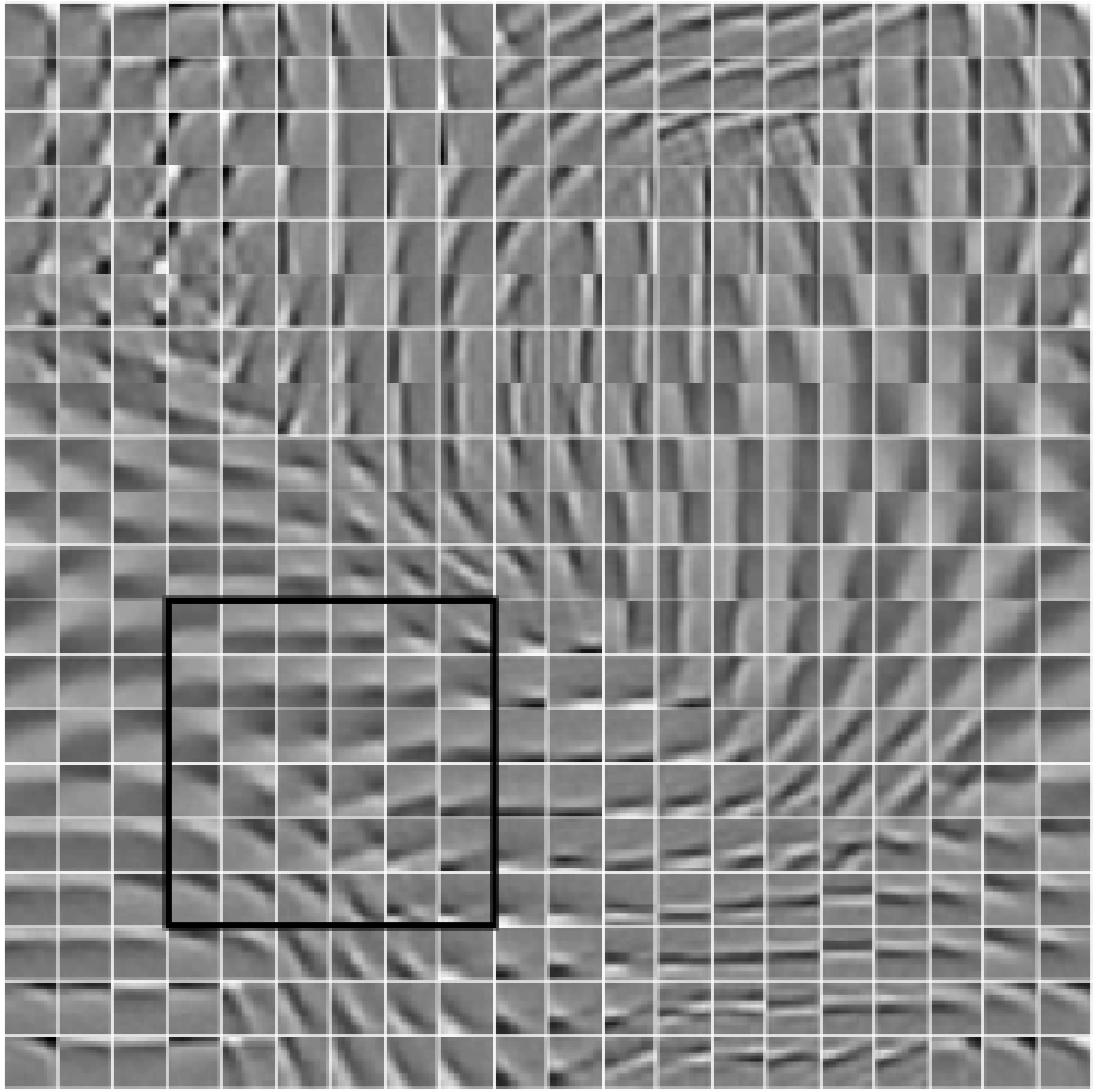


Image-level training, local filters but no weight sharing

- Training on 115x115 images. Kernels are 15x15 (not shared across space!)
  - [Gregor & LeCun 2010]
  - Local receptive fields
  - No shared weights
  - 4x overcomplete
  - L2 pooling
  - Group sparsity over pools

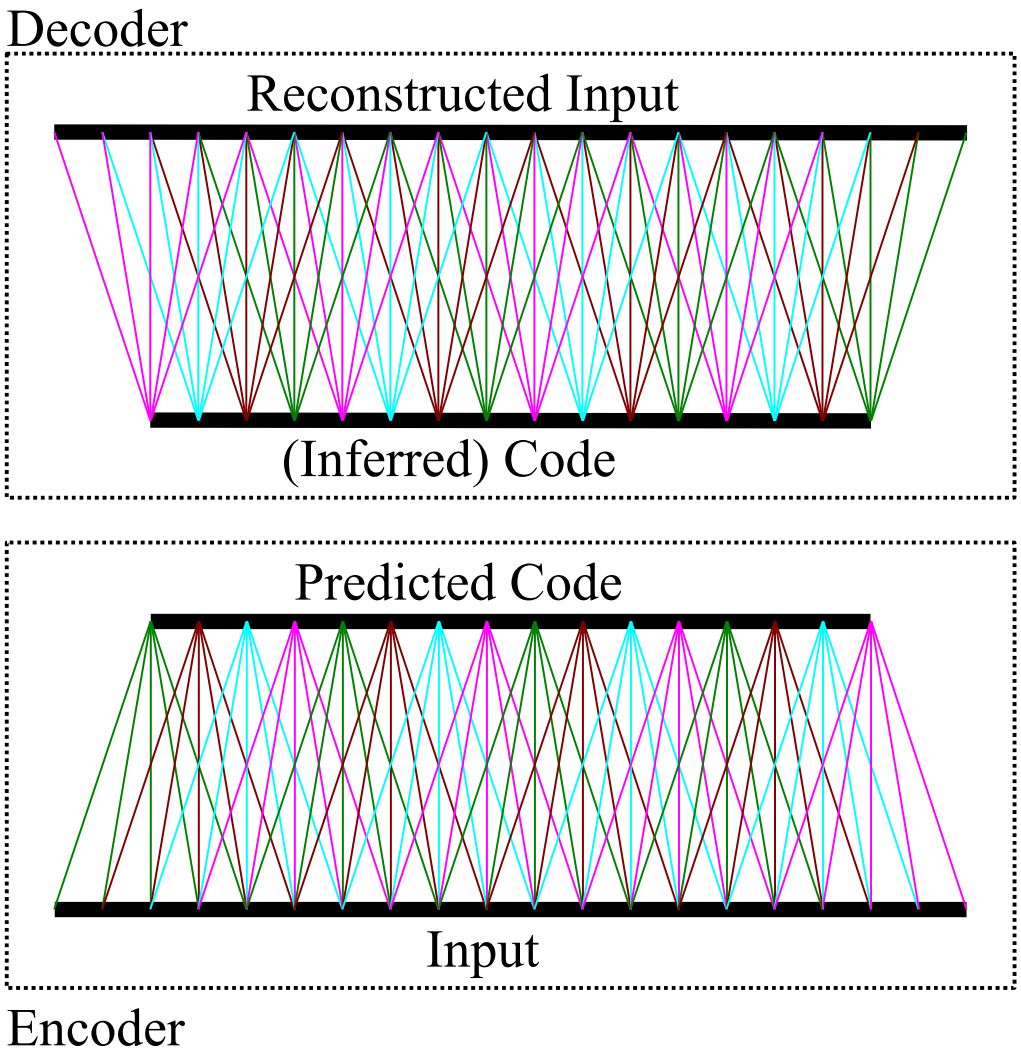
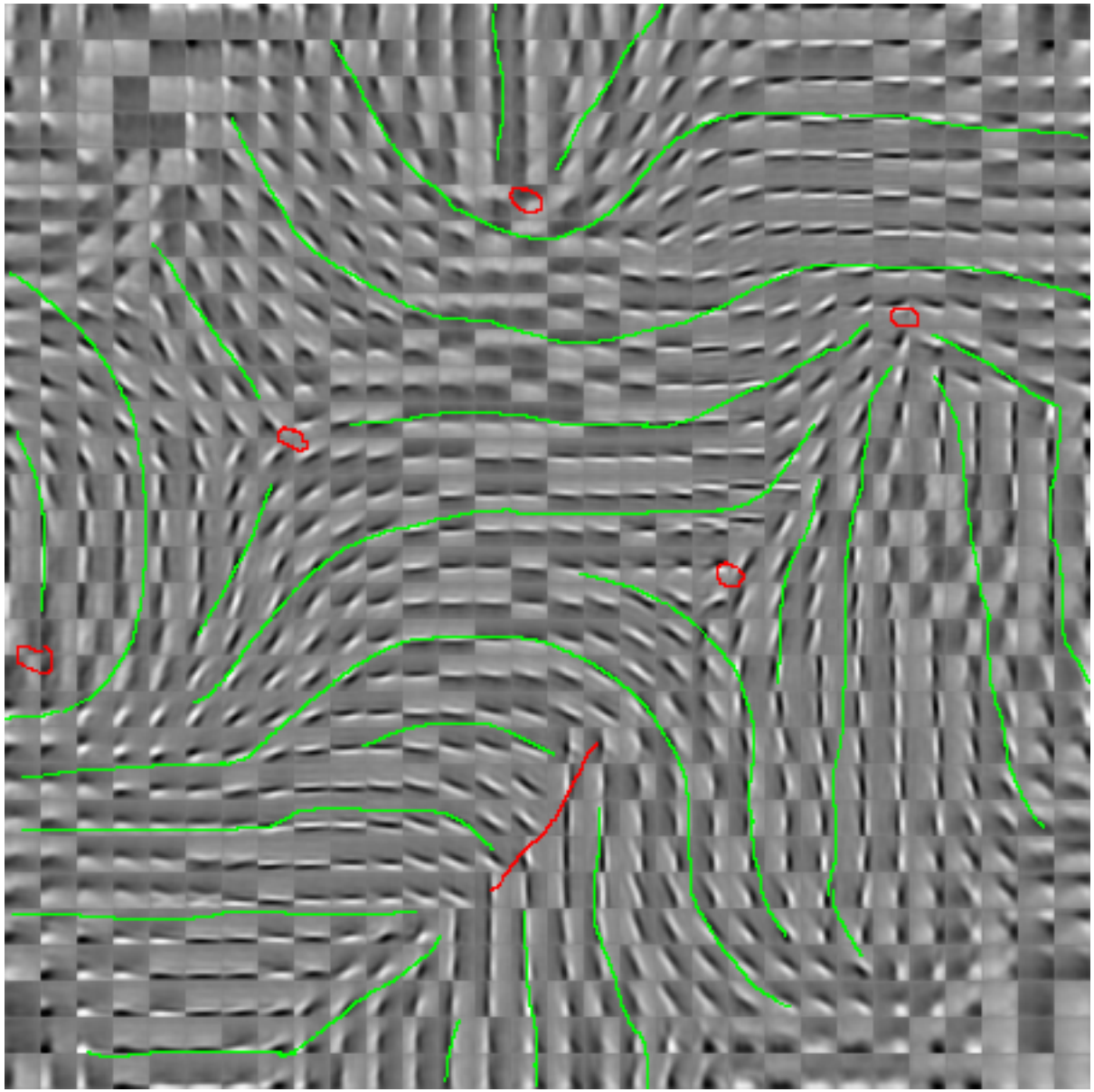


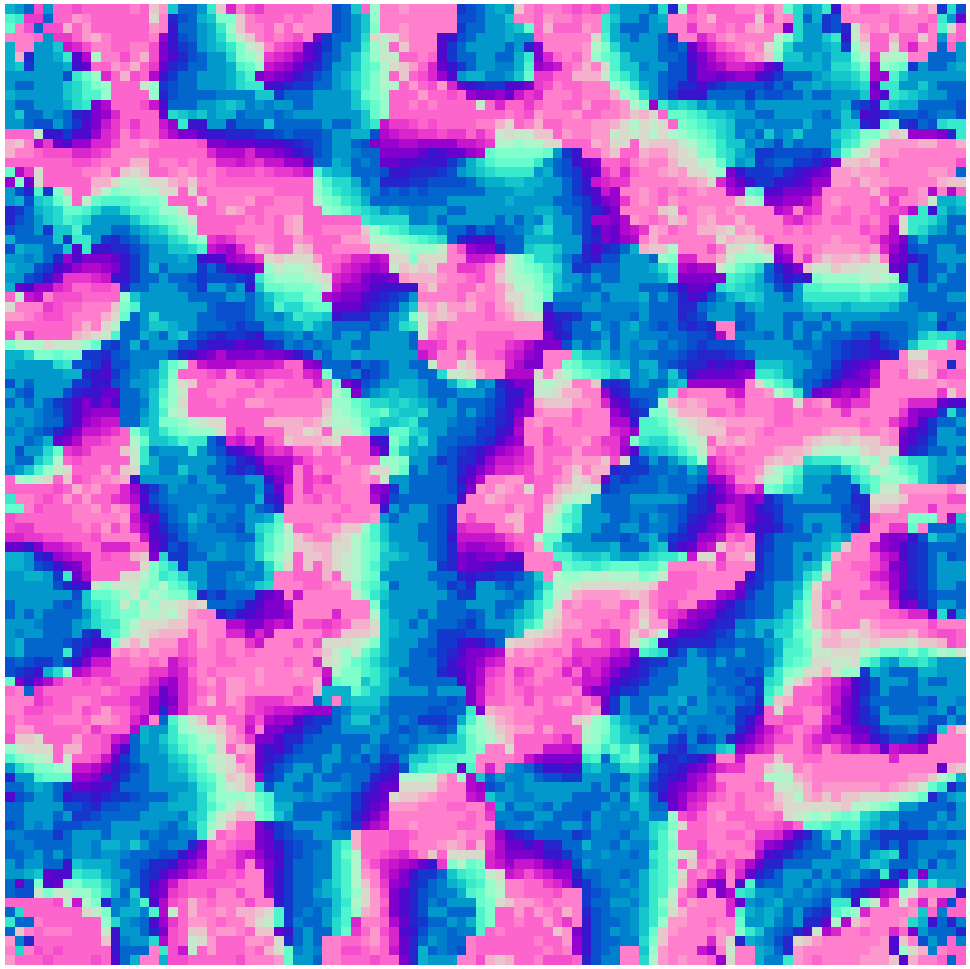
Image-level training, local filters but no weight sharing

- Training on 115x115 images. Kernels are 15x15 (not shared across space!)



Topographic Maps

K Obermayer and GG Blasdel, Journal of Neuroscience, Vol 13, 4114-4129 (**Monkey**)

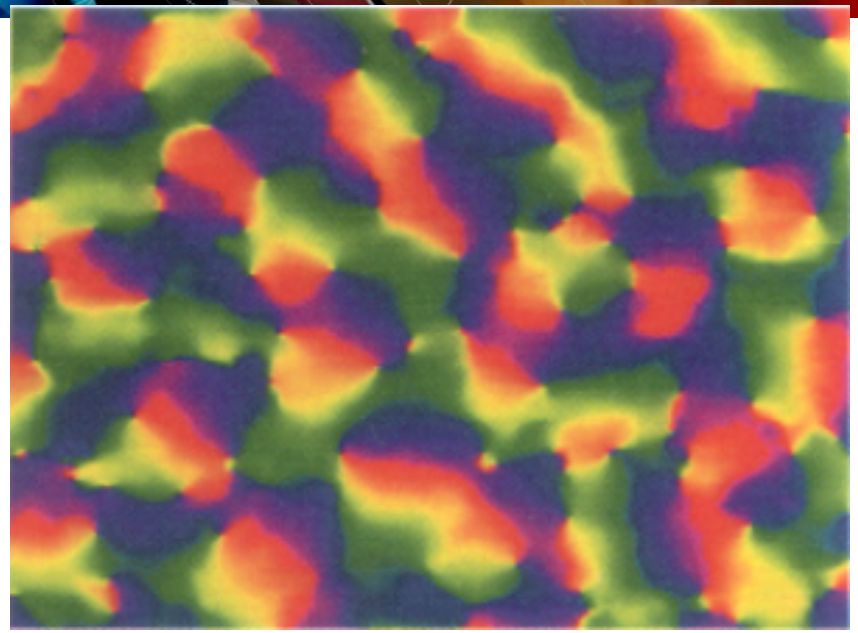


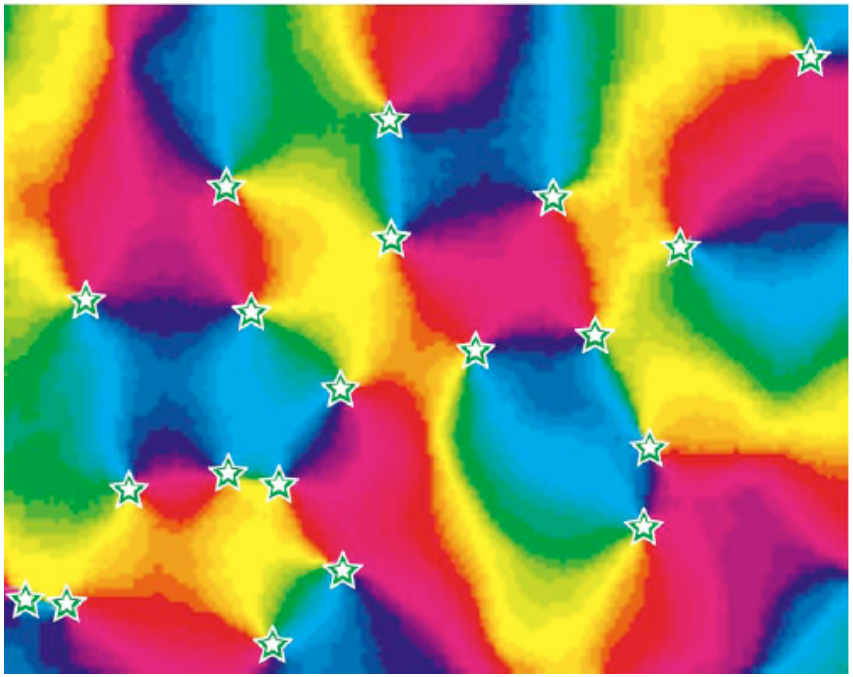
119x119 Image Input

100x100 Code

20x20 Receptive field size

sigma=5

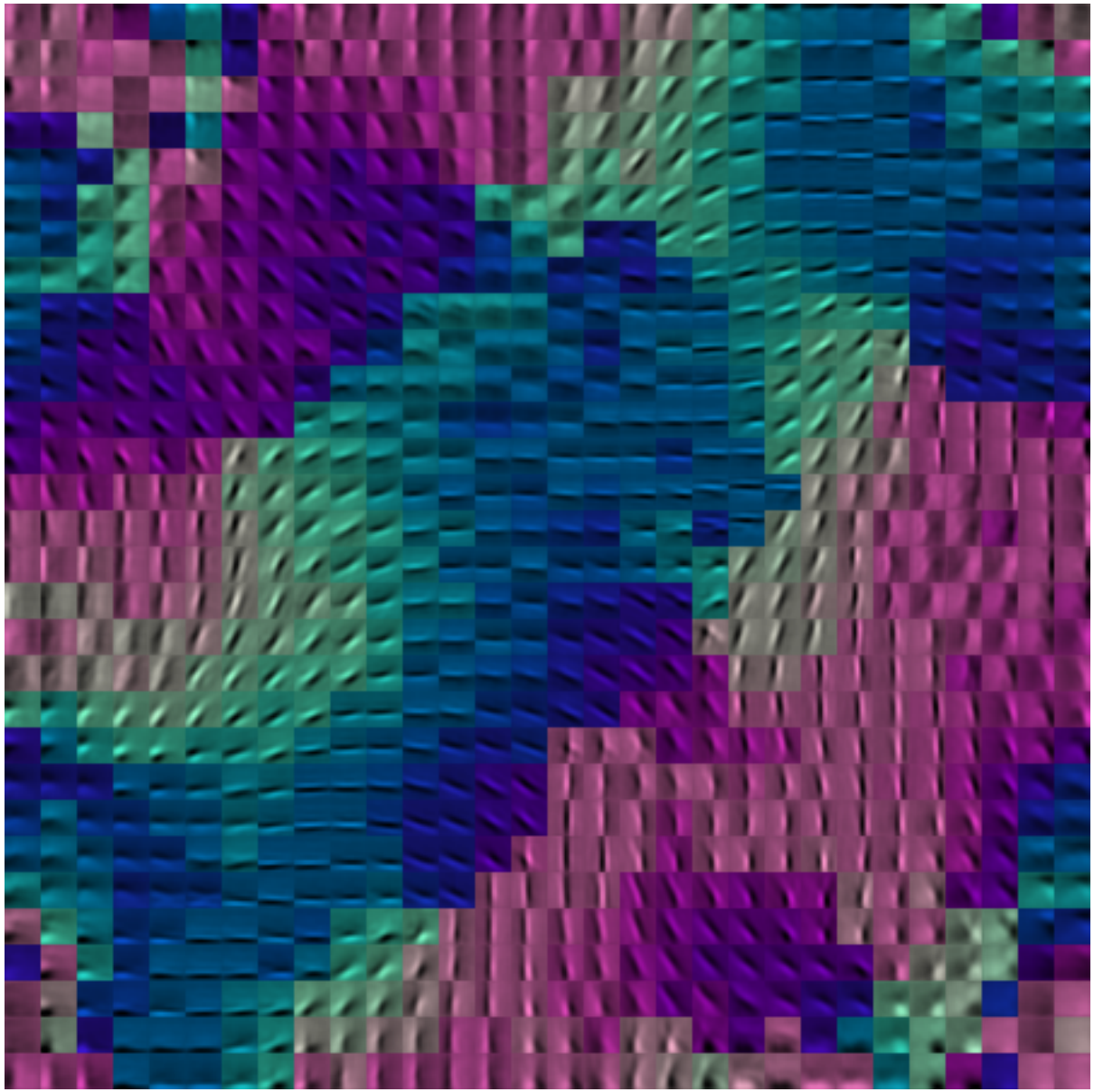




Michael C. Crair, et. al. The Journal of Neurophysiology Vol. 77 No. 6 June 1997, pp. 3381-3385 (**Cat**)

Image-level training, local filters but no weight sharing

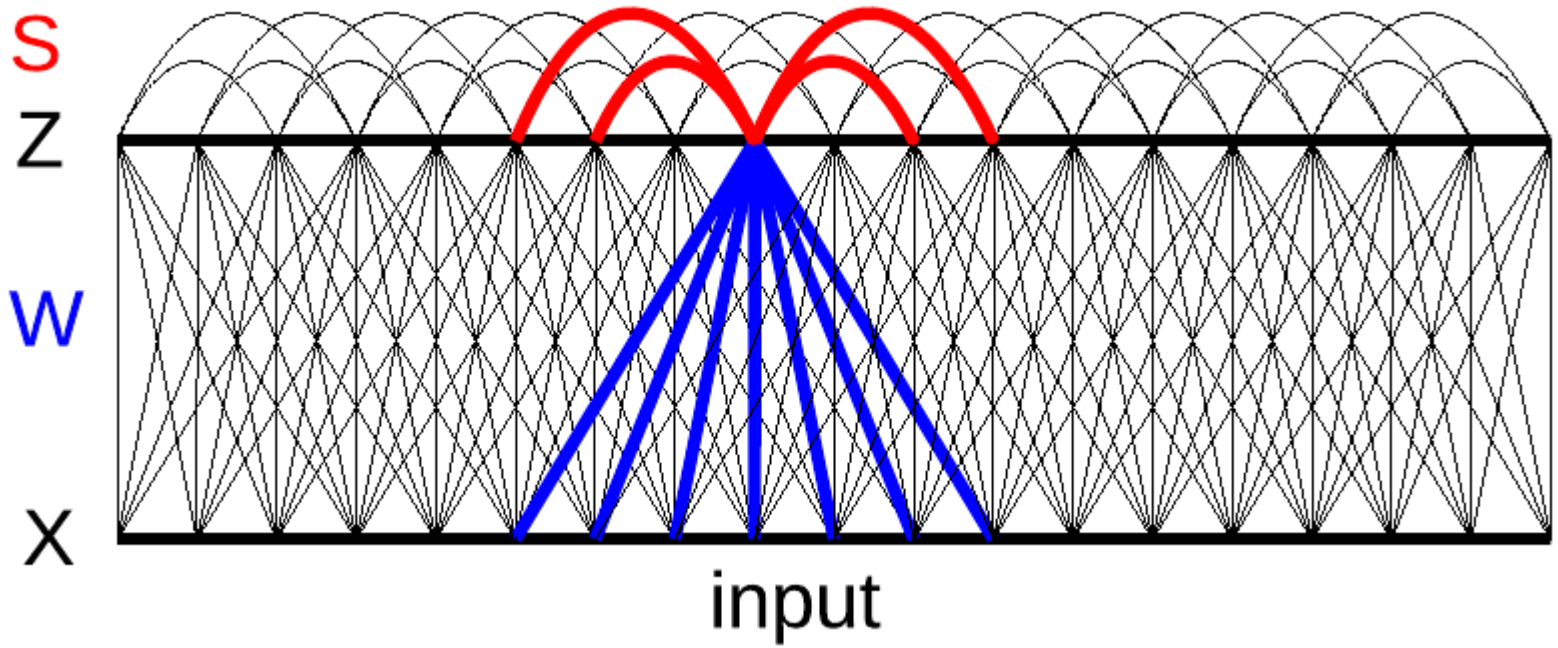
- Color indicates orientation (by fitting Gabors)



Invariant Features Lateral Inhibition

- Replace the L1 sparsity term by a lateral inhibition matrix
- Easy way to impose some structure on the sparsity

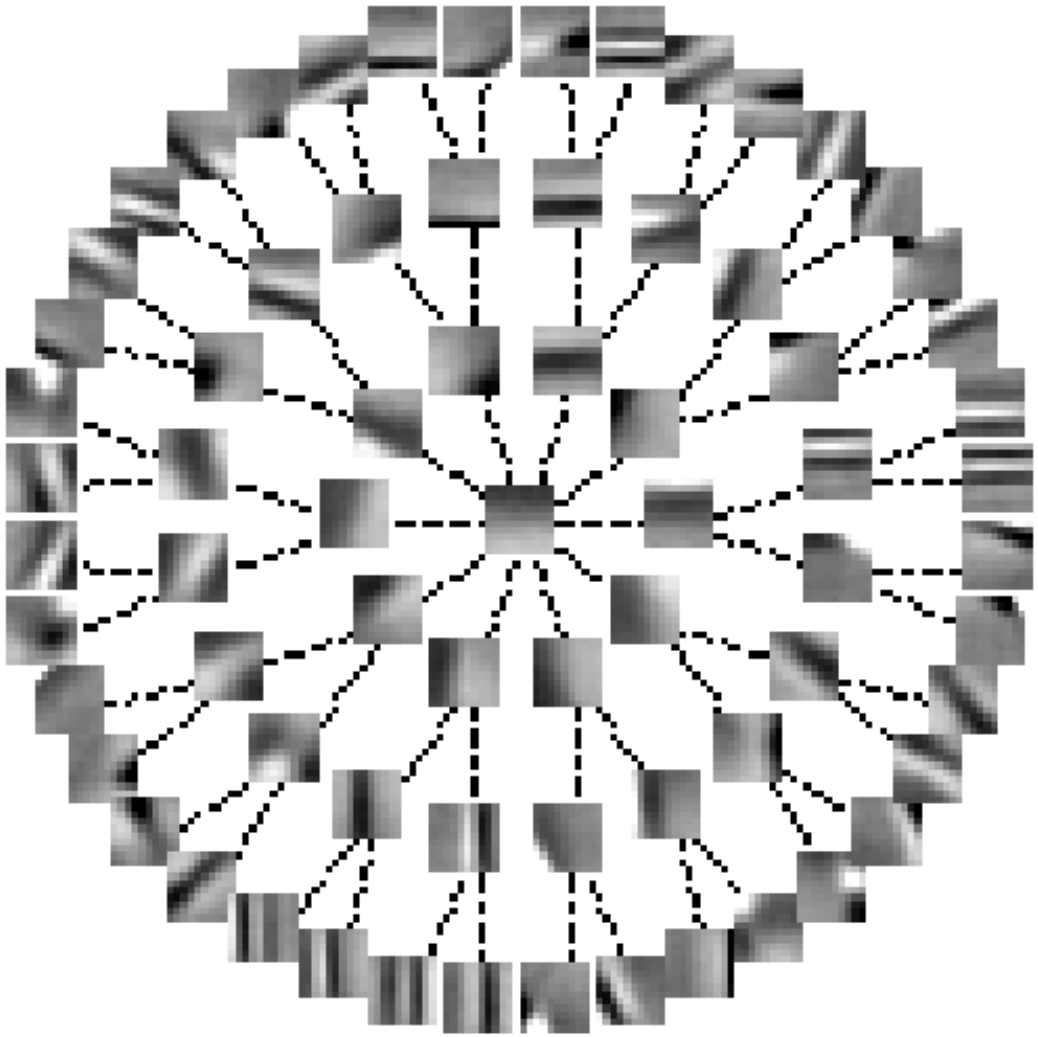
$$\min_{W,Z} \sum_{x \in X} ||Wz - x||^2 + |z|^T S|z|$$



[Gregor, Szlam, LeCun NIPS 2011]

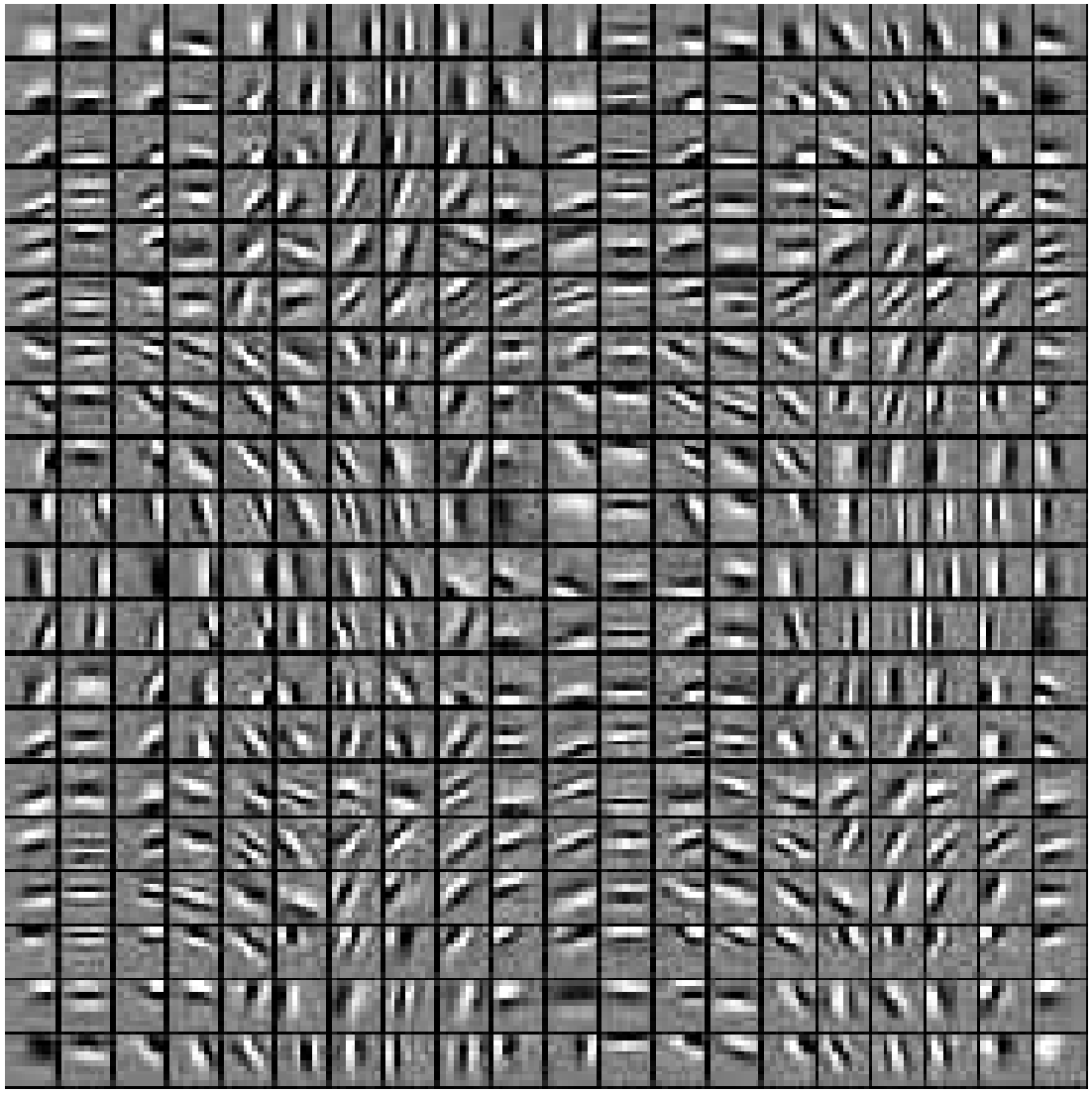
Invariant Features via Lateral Inhibition: Structured Sparsity

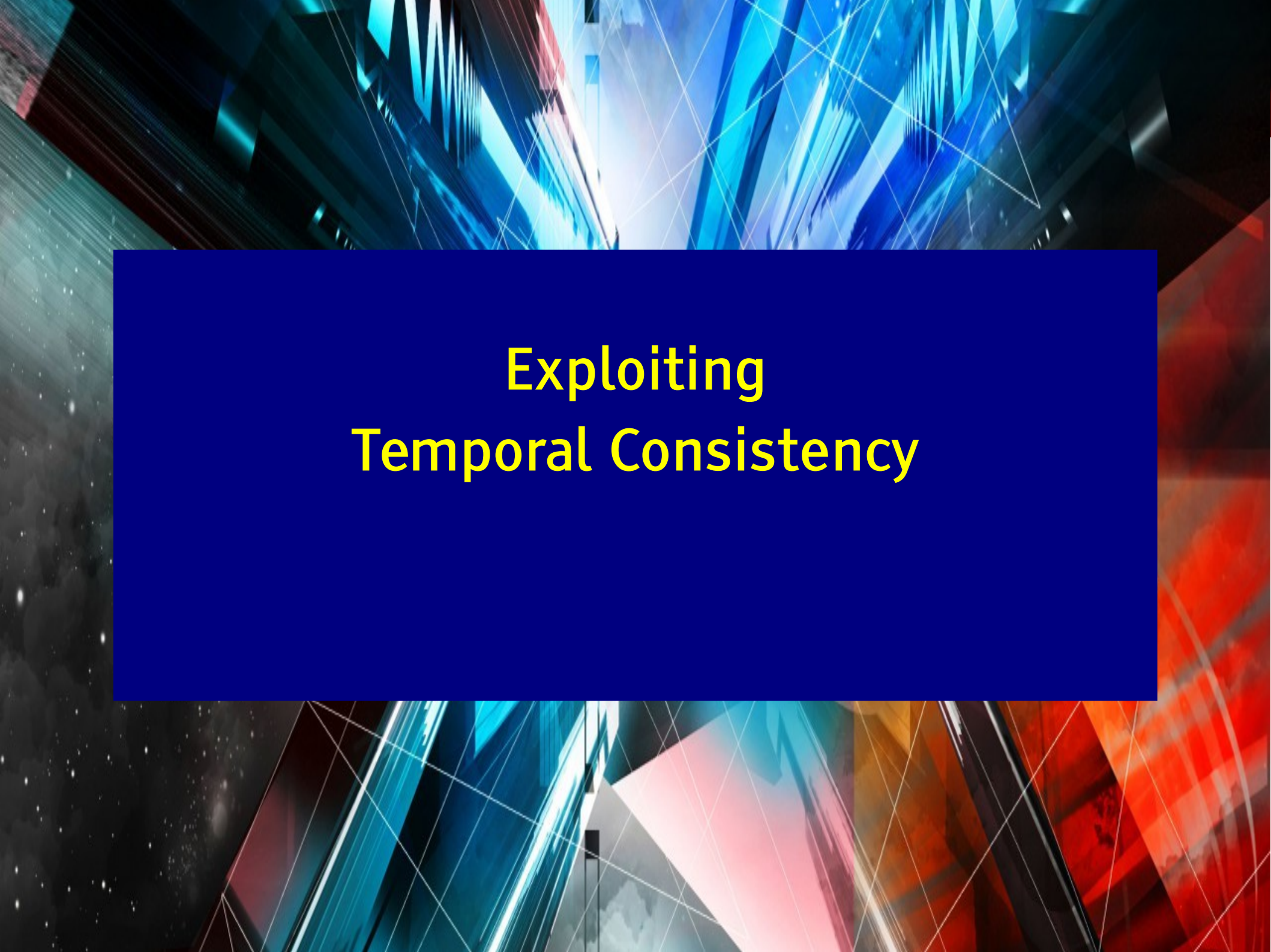
- Each edge in the tree indicates a zero in the S matrix (no mutual inhibition)
- Sij is larger if two neurons are far away in the tree



Invariant Features via Lateral Inhibition: Topographic Maps

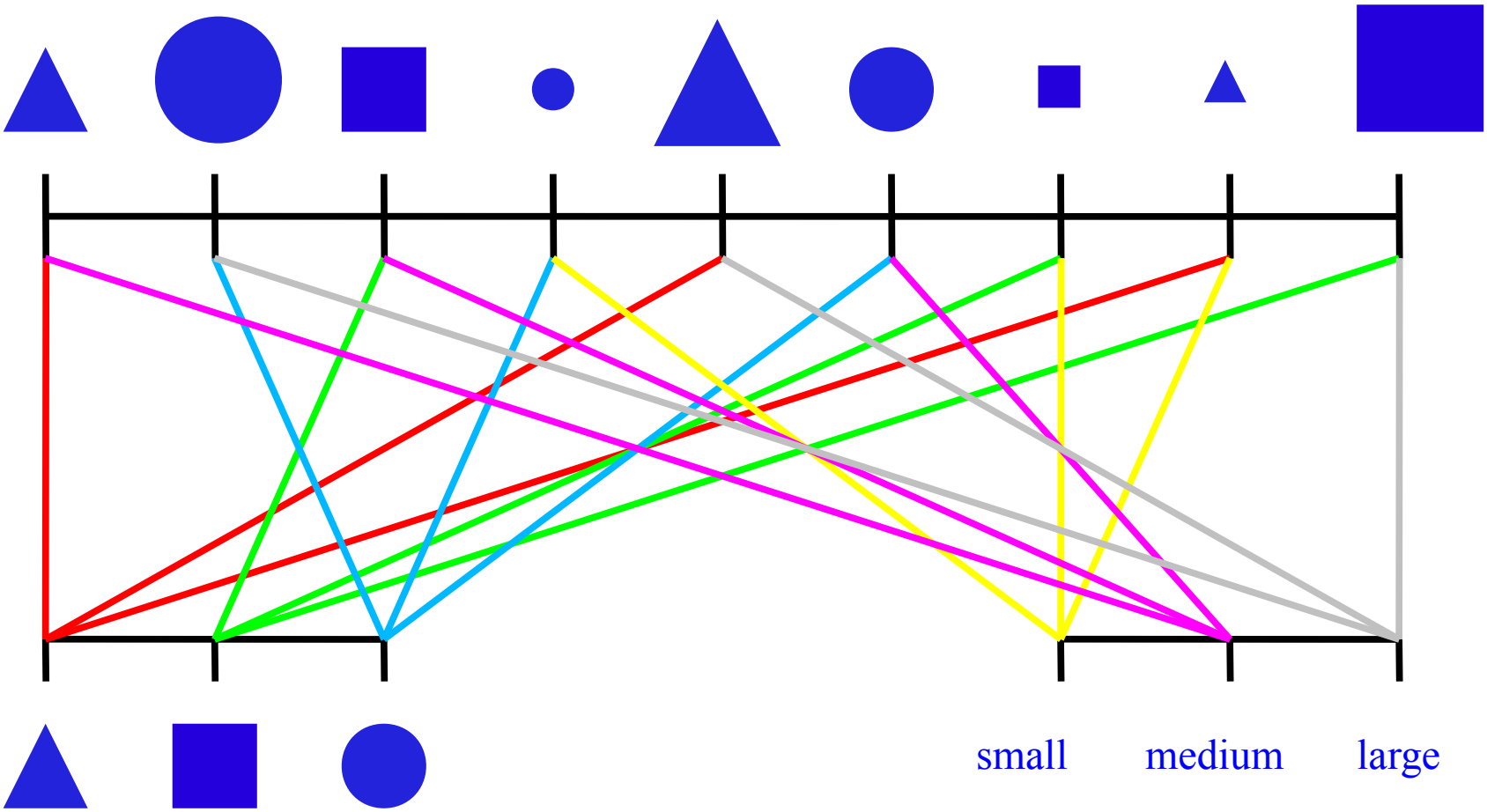
- Non-zero values in S form a ring in a 2D topology
  - Input patches are high-pass filtered




Exploiting
Temporal Consistency

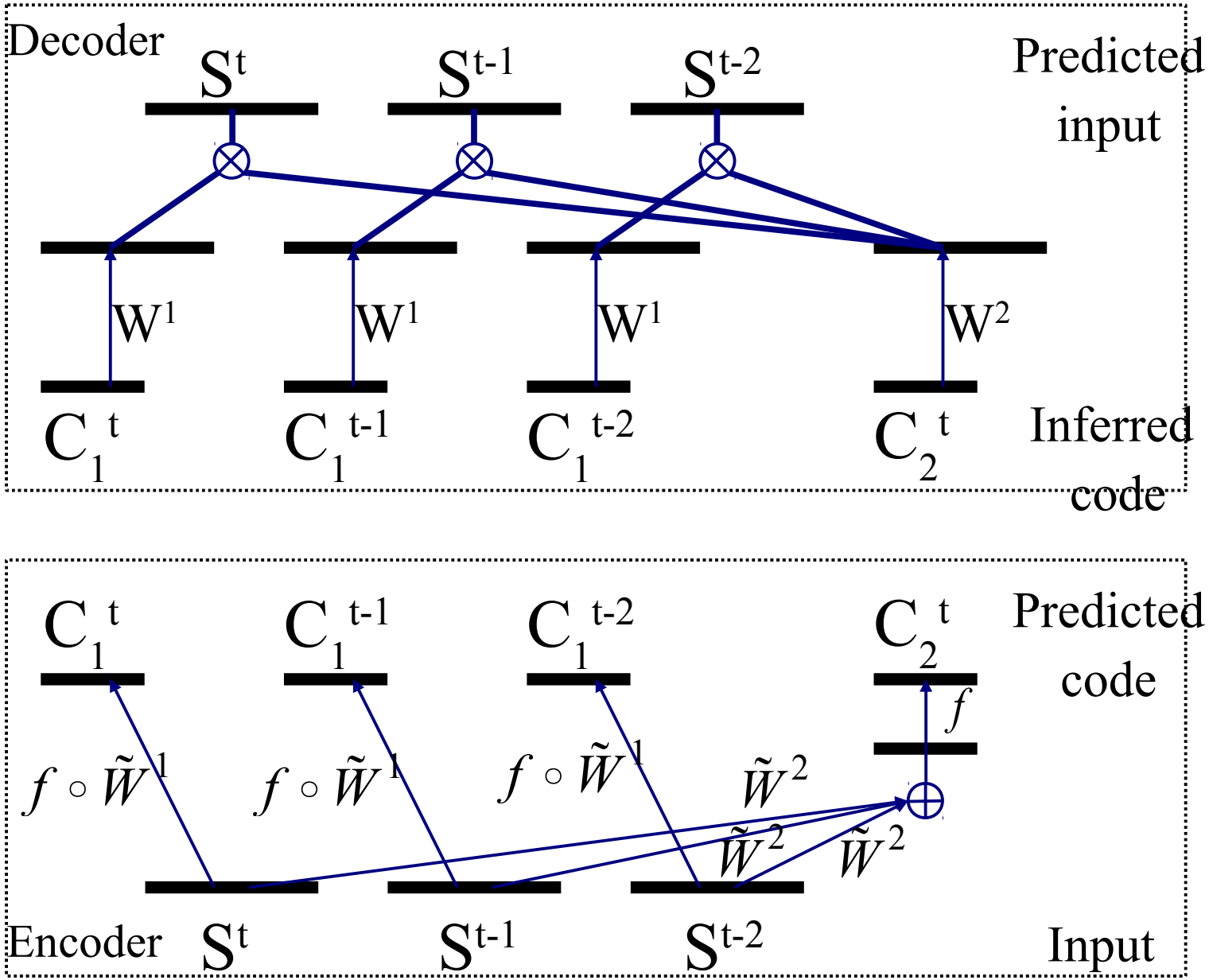
Invariant Features through Temporal Constancy

- Object is cross-product of object type and instantiation parameters
  - Mapping units [Hinton 1981], capsules [Hinton 2011]



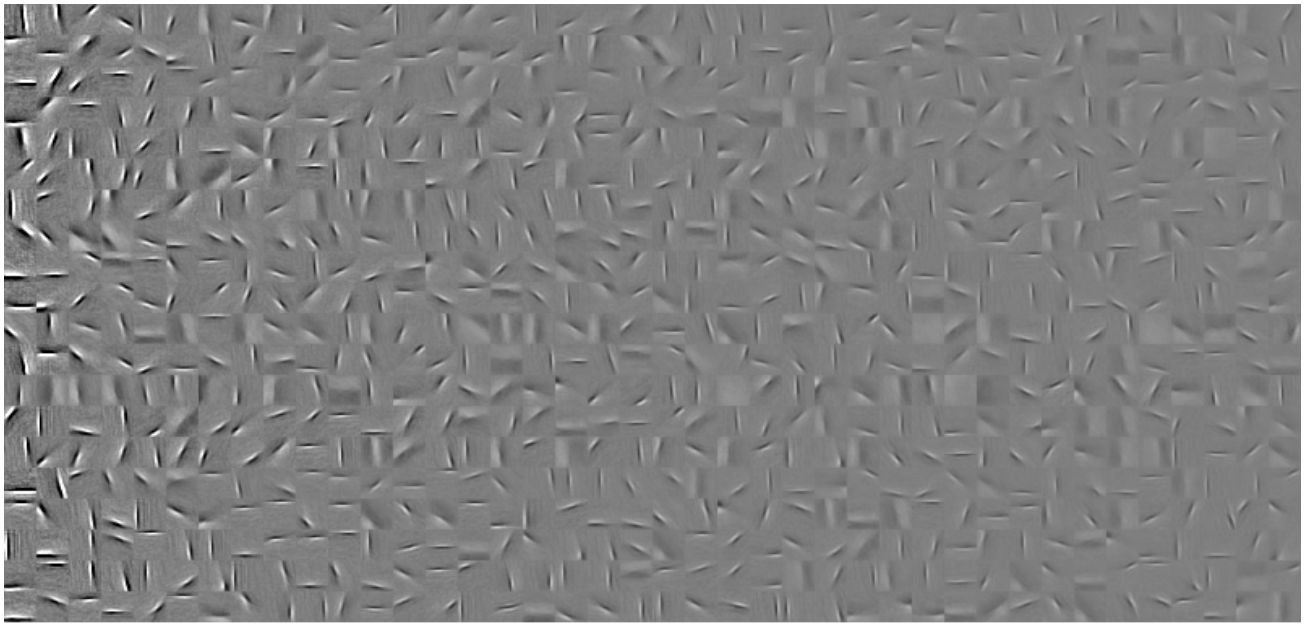
[Karol Gregor et al.]

What-Where Auto-Encoder Architecture

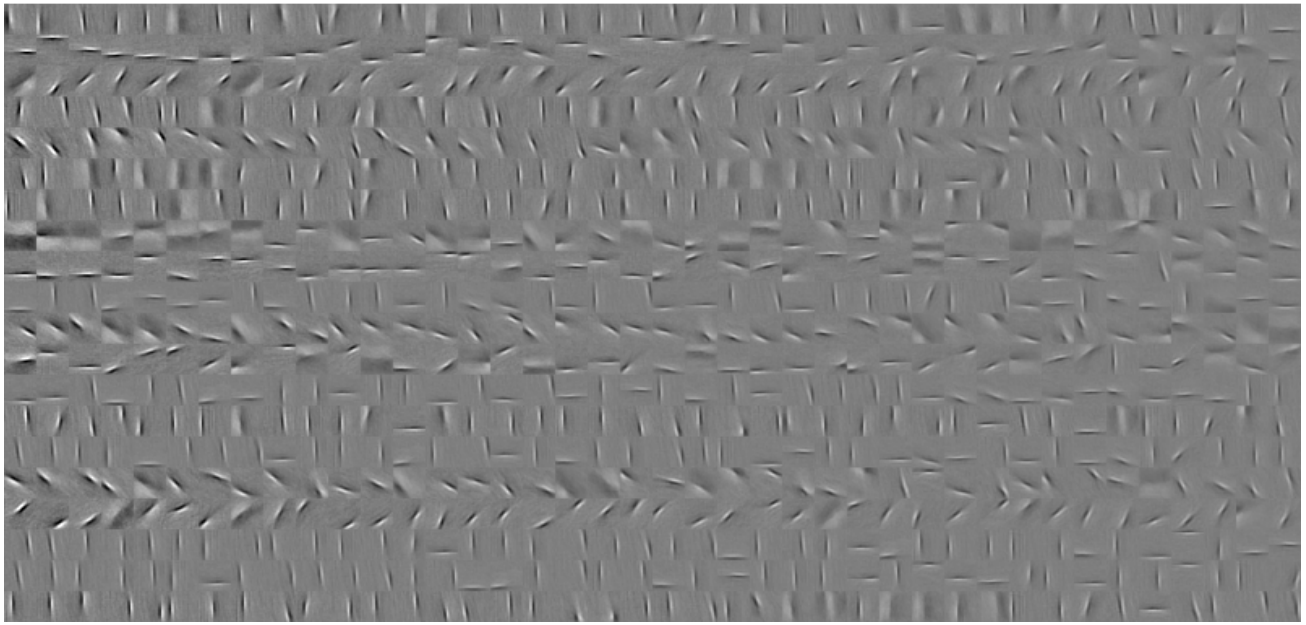


Low-Level Filters Connected to Each Complex Cell

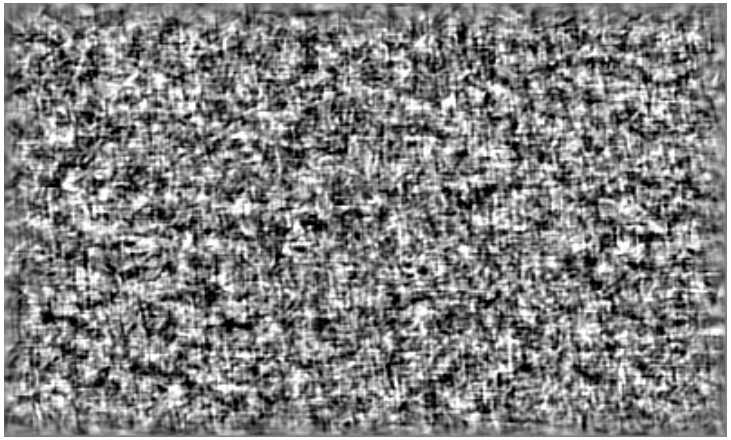
C1
(where)

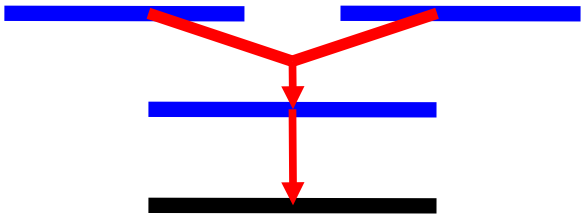


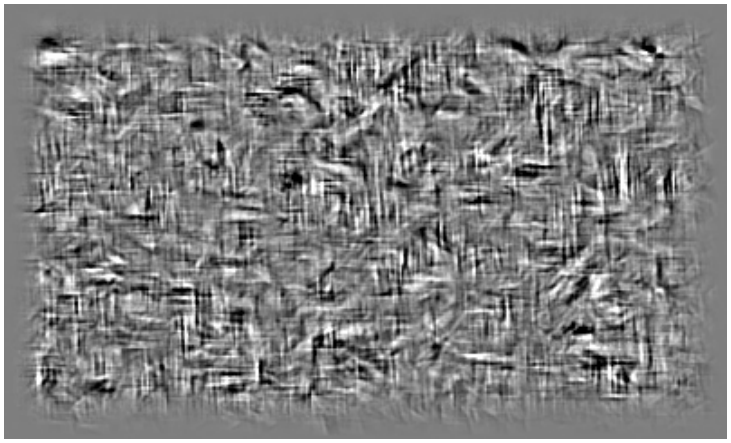
C2
(what)

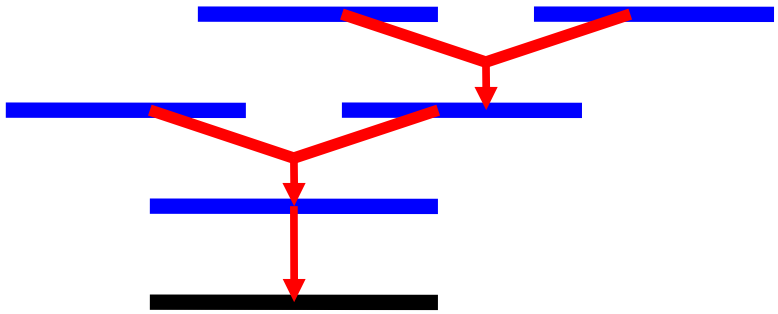


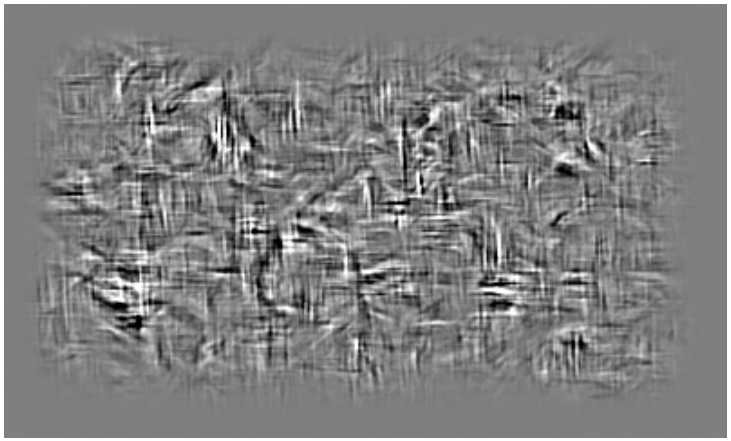
Generating images





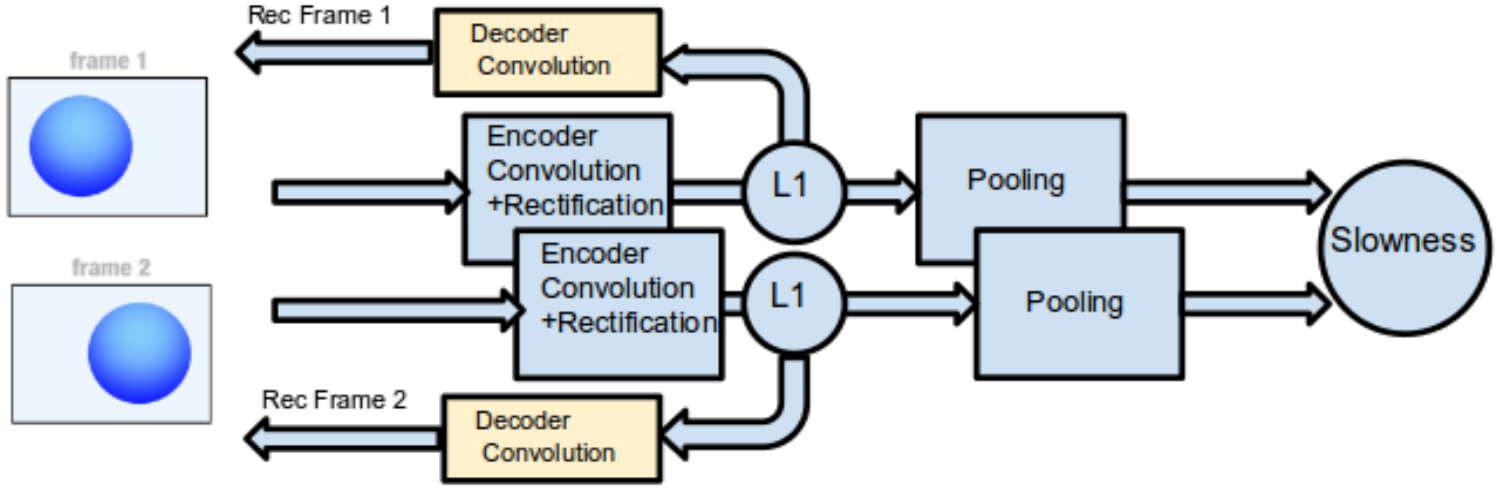


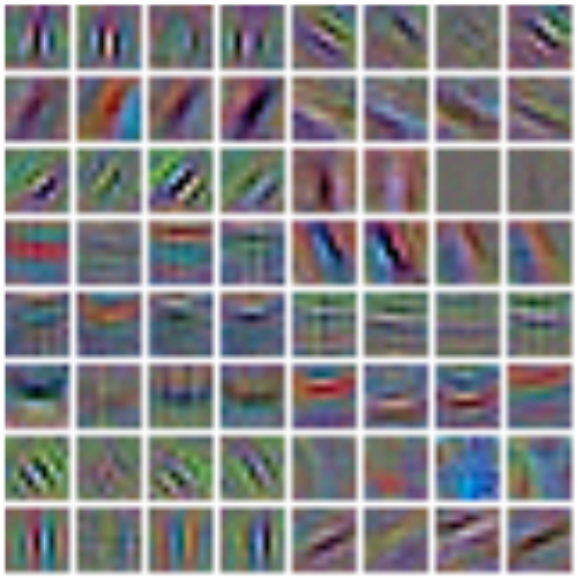




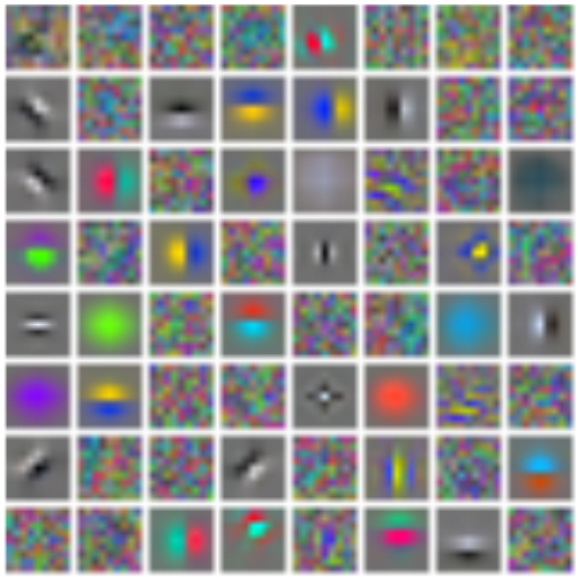
Sparse Auto-Encoder with "Slow Feature" Penalty

Y LeCun

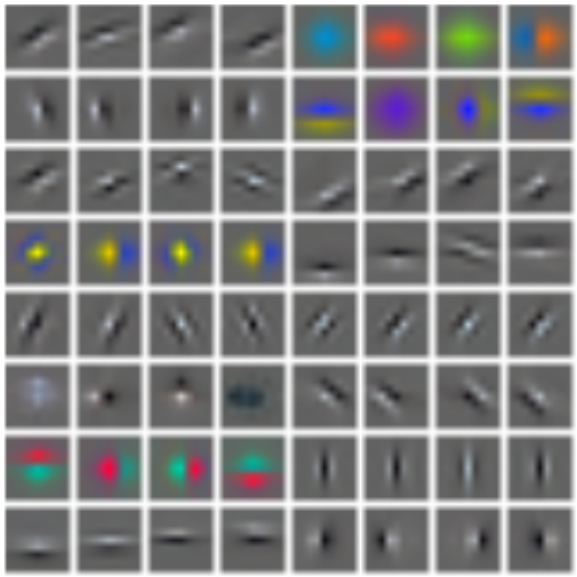




Supervised filters CIFAR10



sparse conv. auto-encoder



slow & sparse conv. auto-encoder

Trained on YouTube videos

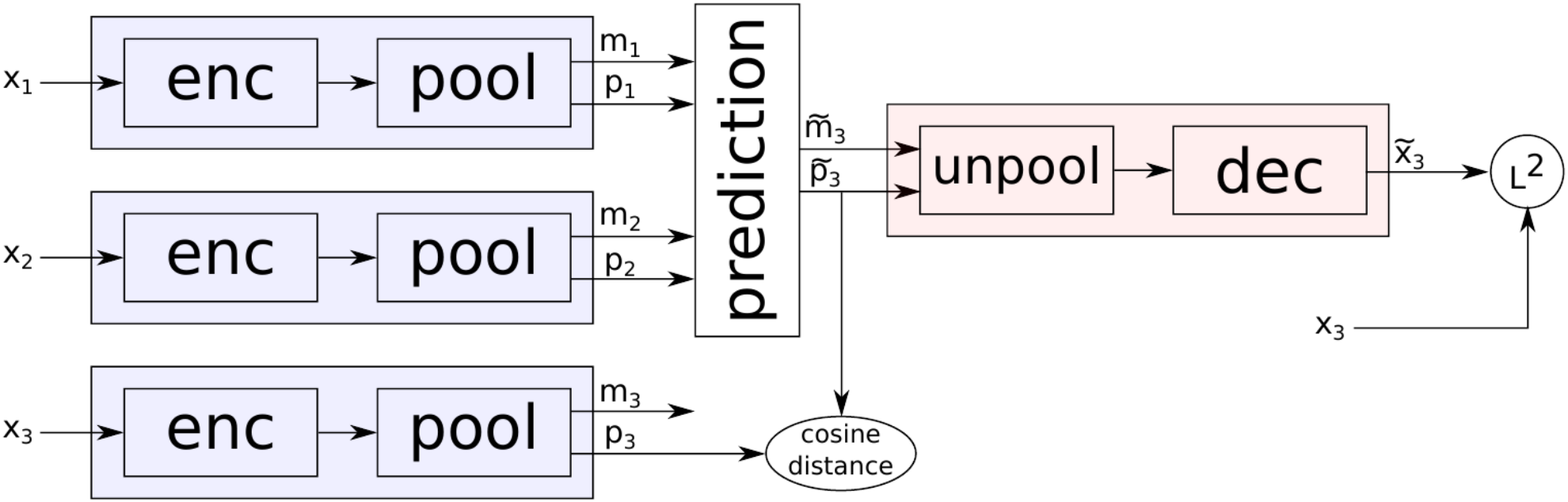
[Goroshin et al. ICCV 2015, Arxiv:1412.6056]

Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun NIPS 2015, arXiv:1506.03011]

- Trained on 3 successive frames of video
- Maximize the colinearity between successive changes of feature vectors
- So that "natural" changes stay in linear manifold in feature space.

$$L = \frac{1}{2}\|G_W(\mathbf{a}\begin{bmatrix} z^t & z^{t-1}\end{bmatrix}^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T(z^{t+1} - z^t)}{\|z^t - z^{t-1}\|\|z^{t+1} - z^t\|}$$



Unsupervised Learning by Prediction and Linearization

Y LeCun

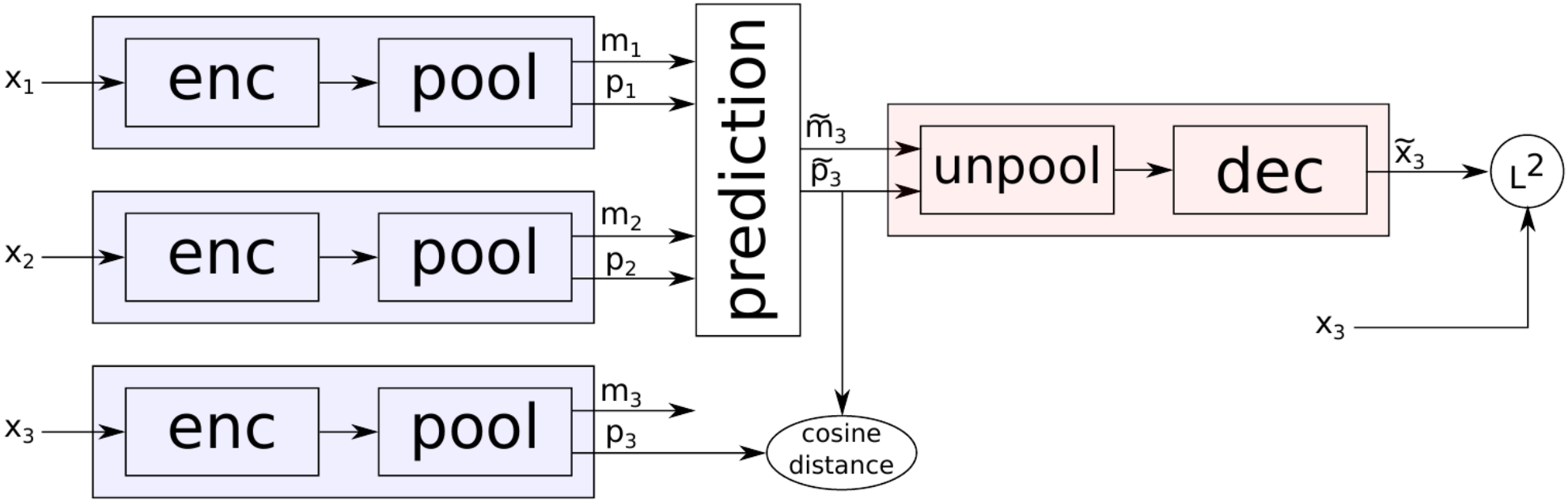
$$L = \frac{1}{2}\|G_W(\mathbf{a}\begin{bmatrix} z^t & z^{t-1}\end{bmatrix}^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T(z^{t+1} - z^t)}{\|z^t - z^{t-1}\|\|z^{t+1} - z^t\|}$$

- **Magnitude**
  - (soft max)

$$m_k = \sum_{N_k} z(f,x,y) \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f\prime,x\prime,y\prime)}} \approx \max_{N_k} z(f,x,y)$$

- **Phase**
  - (soft argmax)

$$\mathbf{p}_k = \sum_{N_k} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f\prime,x\prime,y\prime)}} \approx \arg\max_{N_k} z(f,x,y)$$



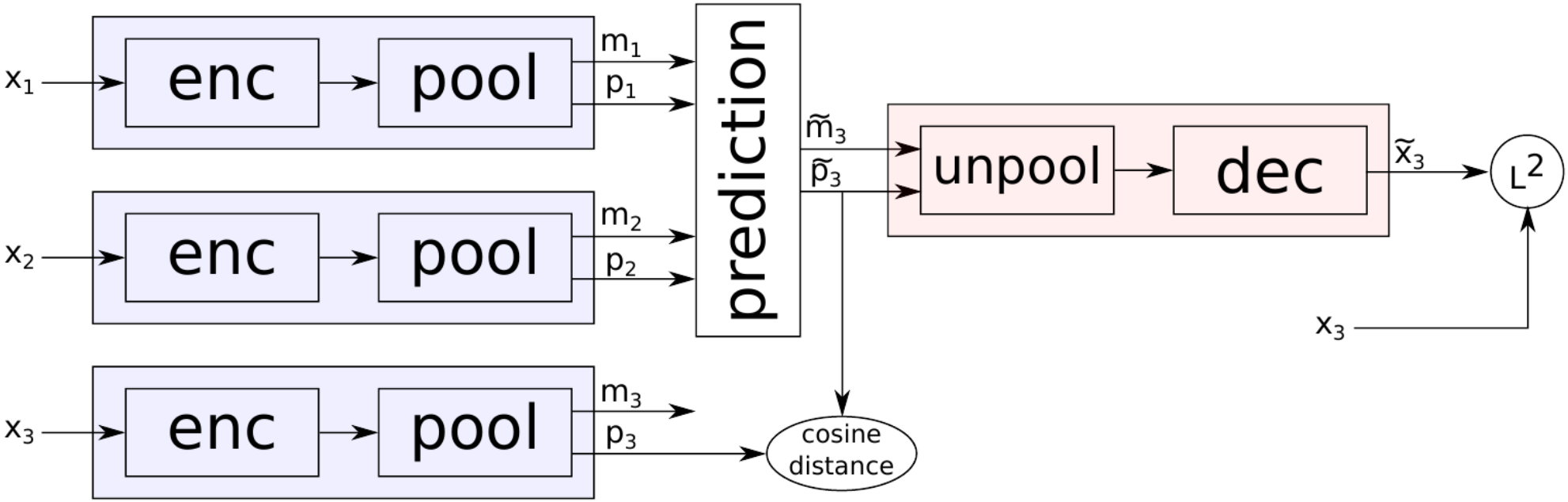
Unsupervised Learning by Prediction and Linearization

[Goroshin, Mathieu, LeCun NIPS 2015, arXiv:1506.03011]

- **Version 2:**
  - Make sure the "what"/magnitude changes slowly
  - Make sure the "where"/phase changes linearly

$$m^{t+1} = \frac{m^t + m^{t-1}}{2}$$

$$\mathbf{p}^{t+1} = 2\mathbf{p}^t - \mathbf{p}^{t-1}$$

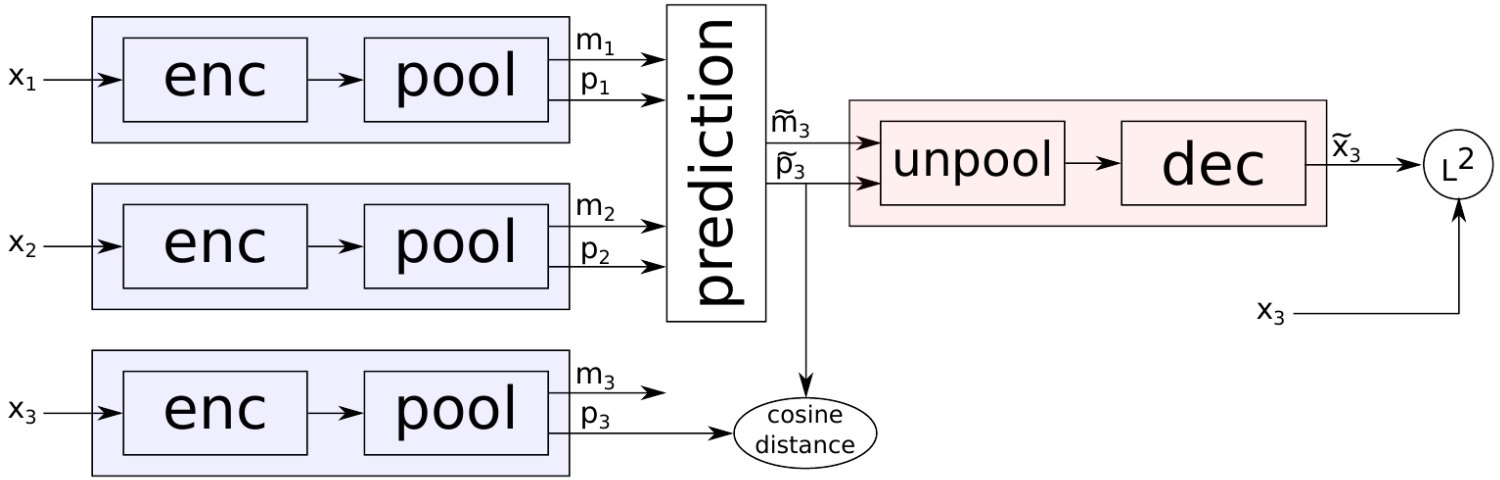


[Goroshin, Mathieu, LeCun arXiv:1506.03011]

- Version 3: the world is unpredictable
  - Add latent variable to compensate for that.
  - Minimize over them during training

$$\hat{z}_\delta^{t+1} = z^t + (W_1\delta) \odot \mathbf{a}\begin{bmatrix} z^t & z^{t-1} \end{bmatrix}^T$$

$$L = \min_\delta \|G_W(\hat{z}_\delta^{t+1}) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T(z^{t+1} - z^t)}{\|z^t - z^{t-1}\|\|z^{t+1} - z^t\|}$$



Algorithm (with latent variables)

Algorithm 1 Minibatch stochastic gradient descent training for prediction with uncertainty. The number of δ-gradient descent steps (k) is treated as a hyper-parameter.

for number of training epochs **do**
  Sample a mini-batch of temporal triplets $\{x^{t-1}, x^t, x^{t+1}\}$
  Set $\delta_0 = 0$
  Forward propagate x^{t-1}, x^t through the network and obtain the codes z^{t-1}, z^t and the prediction $\hat{x}_0^{t+1}$
  for i =1 to k **do**
    Compute the L^2 prediction error
    Back propagate the error through the decoder to compute the gradient $\frac{\partial L}{\partial \delta^{i-1}}$
    Update $\delta_i = \delta_{i-1} - \alpha \frac{\partial L}{\partial \delta^{i-1}}$
    Compute $\hat{z}_{\delta_i}^{t+1} = z^t + (W_1 \delta_i) \odot \mathbf{a} \begin{bmatrix} z^t & z^{t-1} \end{bmatrix}^T$
    Compute $\hat{x}_i^{t+1} = G_W(z_{\delta^i}^{t+1})$
  end for
  Back propagate the full encoder/predictor loss from Equation 7 using δ_k, and update the weight matrices W and W_1
end for

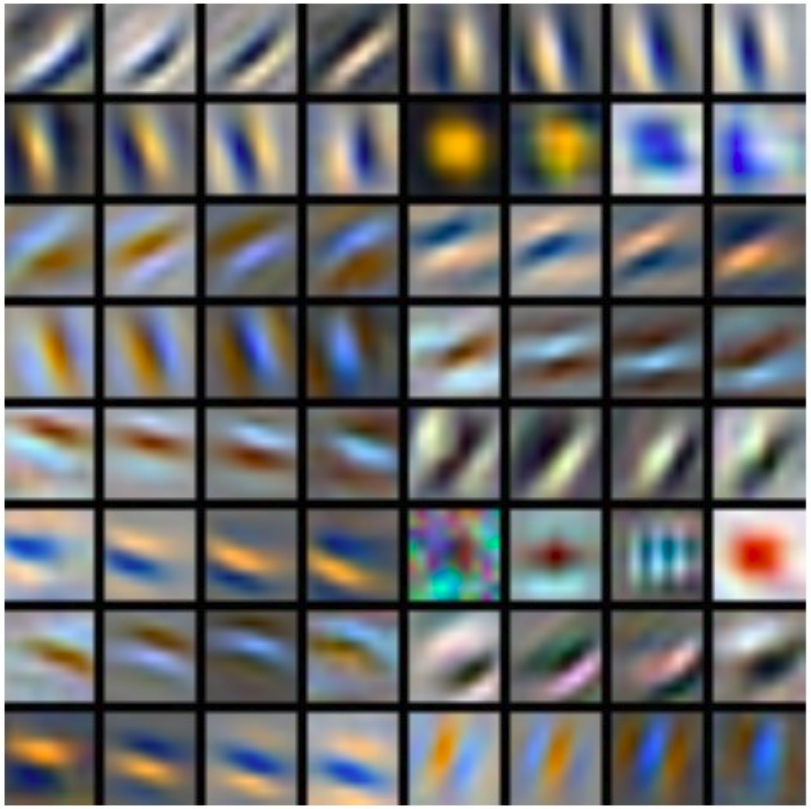
$$\hat{z}_{\delta}^{t+1} = z^t + (W_1 \delta) \odot \mathbf{a} \begin{bmatrix} z^t & z^{t-1} \end{bmatrix}^T$$

$$L = \min_{\delta} \| G_W(\hat{z}_{\delta}^{t+1}) - x^{t+1} \|_2^2 - \lambda \frac{(z^t - z^{t-1})^T (z^{t+1} - z^t)}{\| z^t - z^{t-1} \| \| z^{t+1} - z^t \|}$$

Architectures

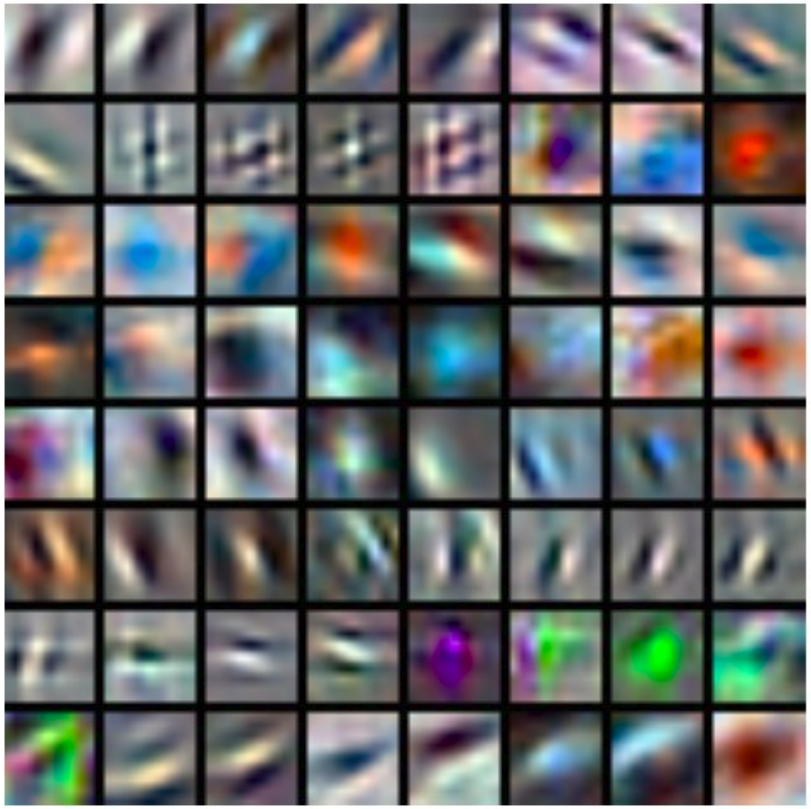
	Encoder	Prediction	Decoder
Shallow Architecture 1	Conv+ReLU $64 \times 9 \times 9$ Phase Pool 4	Average Mag. Linear Extrap. Phase	Conv $64 \times 9 \times 9$
Shallow Architecture 2	Conv+ReLU $64 \times 9 \times 9$ Phase Pool 4 stride 2	Average Mag. Linear Extrap. Phase	Conv $64 \times 9 \times 9$
Deep Architecture 1	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU 8192×4096	None	FC+ReLU $\mathbf{8192} \times 8192$ Reshape $32 \times 16 \times 16$ SpatialPadding 8×8 Conv+ReLU $16 \times 9 \times 9$ SpatialPadding 8×8 Conv $1 \times 9 \times 9$
Deep Architecture 2	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU 8192×4096	Linear Extrapolation	FC+ReLU 4096×8192 Reshape $32 \times 16 \times 16$ SpatialPadding 8×8 Conv+ReLU $16 \times 9 \times 9$ SpatialPadding 8×8 Conv $1 \times 9 \times 9$
Deep Architecture 3	Conv+ReLU $16 \times 9 \times 9$ Conv+ReLU $32 \times 9 \times 9$ FC+ReLU 8192×4096 Reshape $64 \times 8 \times 8$ Phase Pool 8×8	Average Mag. Linear Extrap. Phase	Unpool 8×8 FC+ReLU 4096×8192 Reshape $32 \times 16 \times 16$ SpatialPadding 8×8 Conv+ReLU $16 \times 9 \times 9$ SpatialPadding 8×8 Conv $1 \times 9 \times 9$

Filters: pooling over groups of 4 filters.



(a) Shallow Architecture 1

Non-overlapping pooling
4x4x4 (feat,x,y)



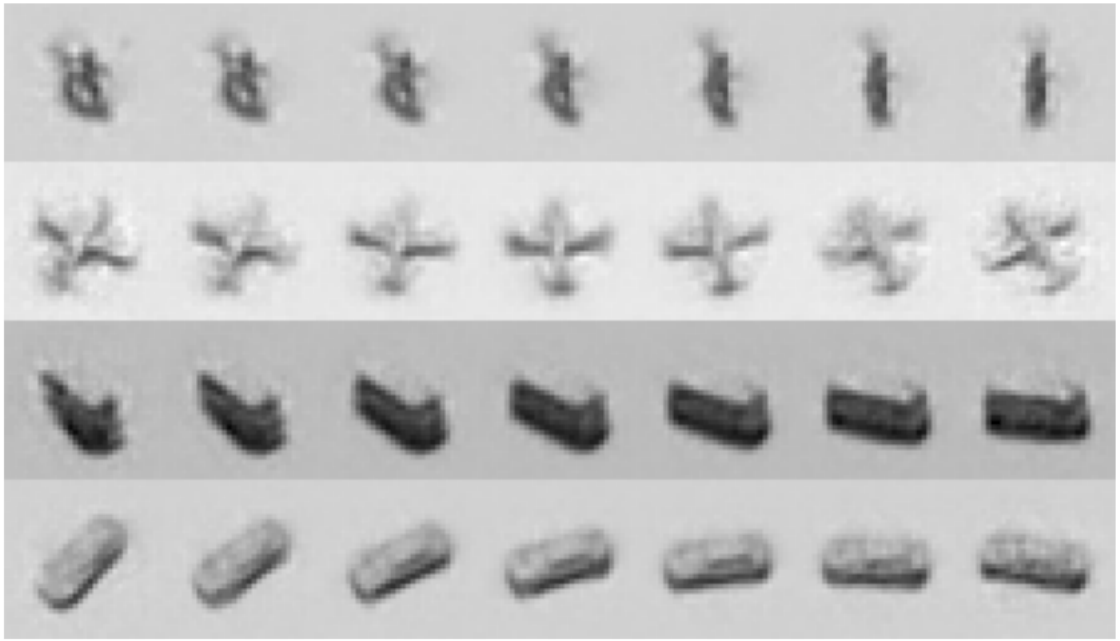
(b) Shallow Architecture 2

Overlapping pooling
4x4x4 (feat,x,y)
Stride 2 over features

Image interpolation in feature space: deterministic world

Y LeCun

- No phase pooling
- No curvature regularization



- With phase pooling
- With curvature regularization

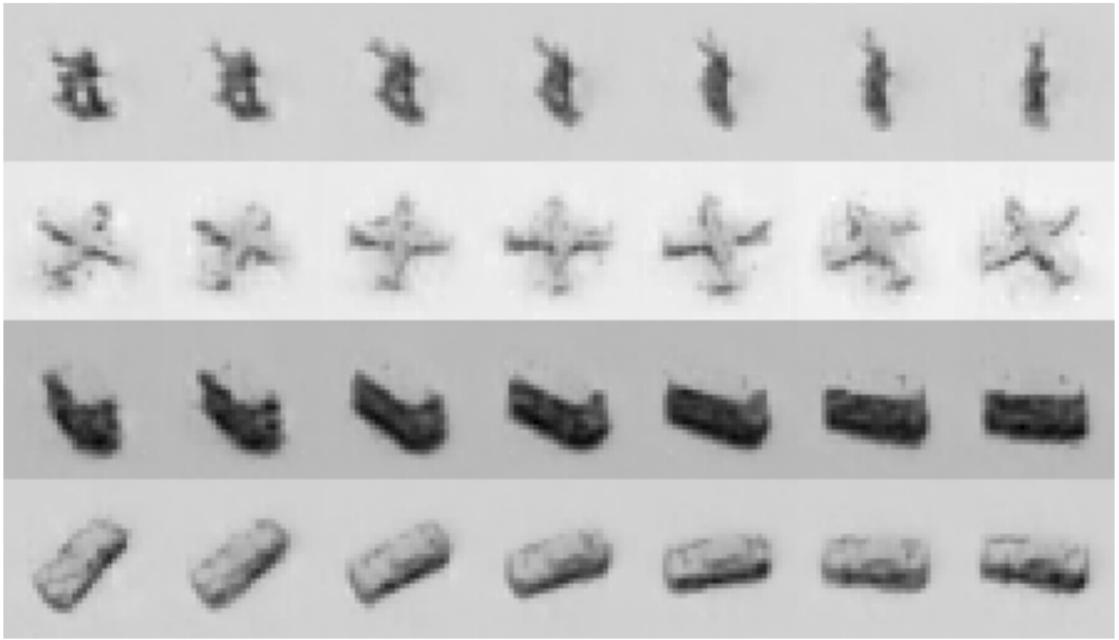


Image interpolation in feature space: deterministic world

Y LeCun

- Image interpolation with the basic model (siamese net)

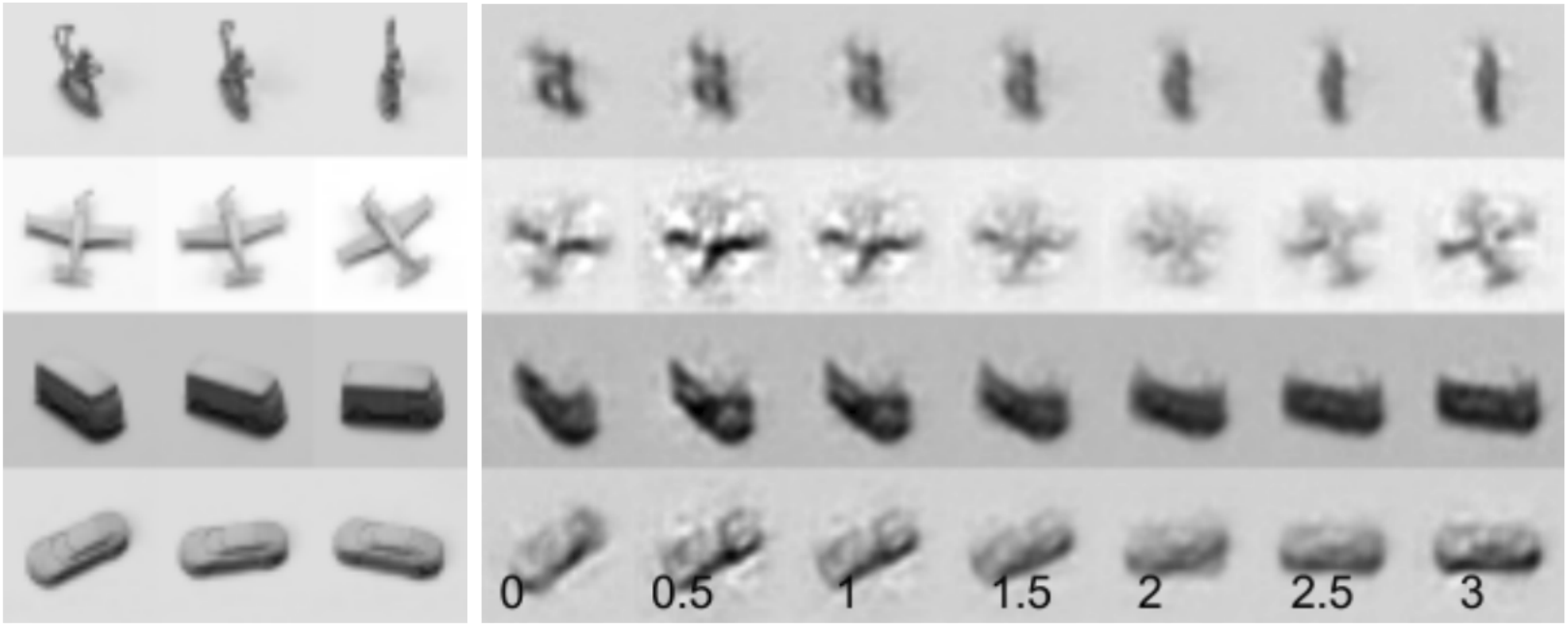
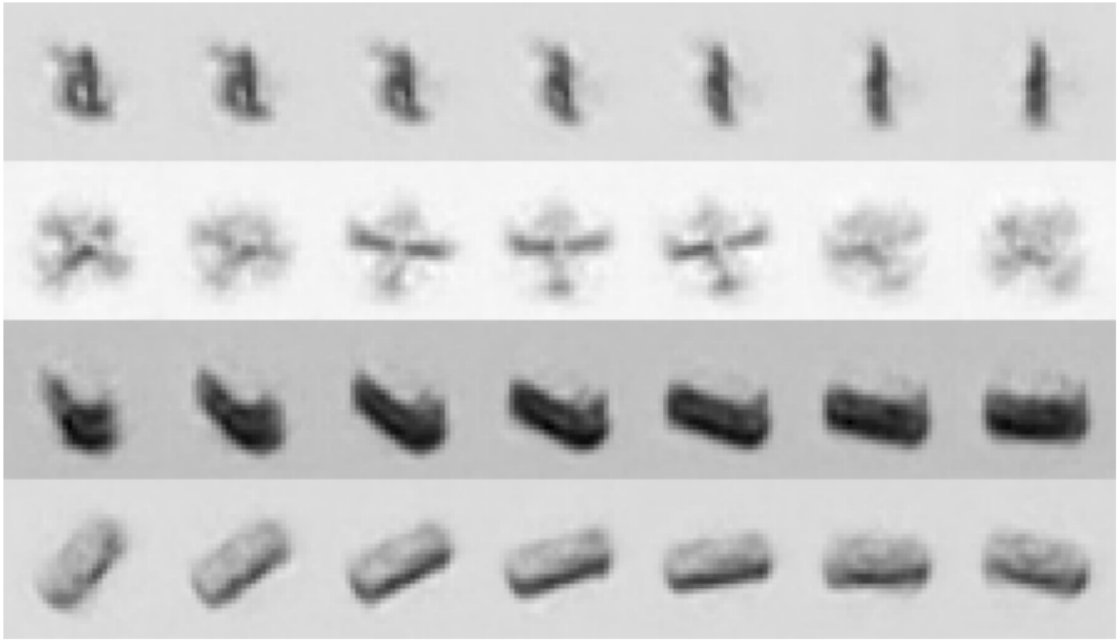


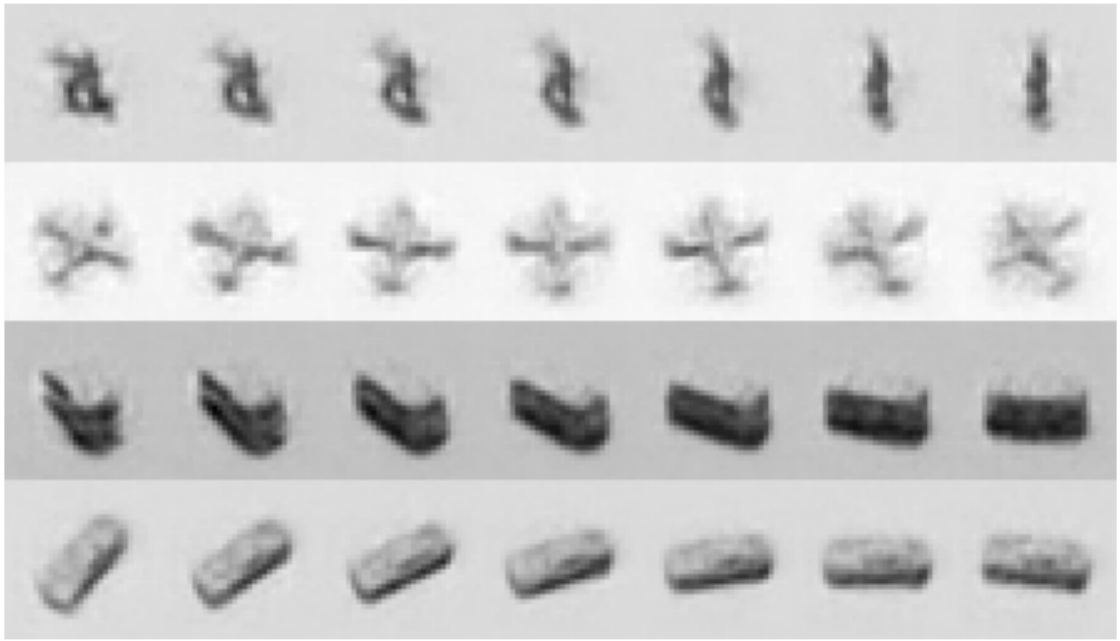
Image interpolation in feature space: unpredictable world

Y LeCun

- Phase interpolation
- No latent variable



- Phase interpolation
- With latent variable



Unsupervised Learning by Prediction and Linearization

Y LeCun

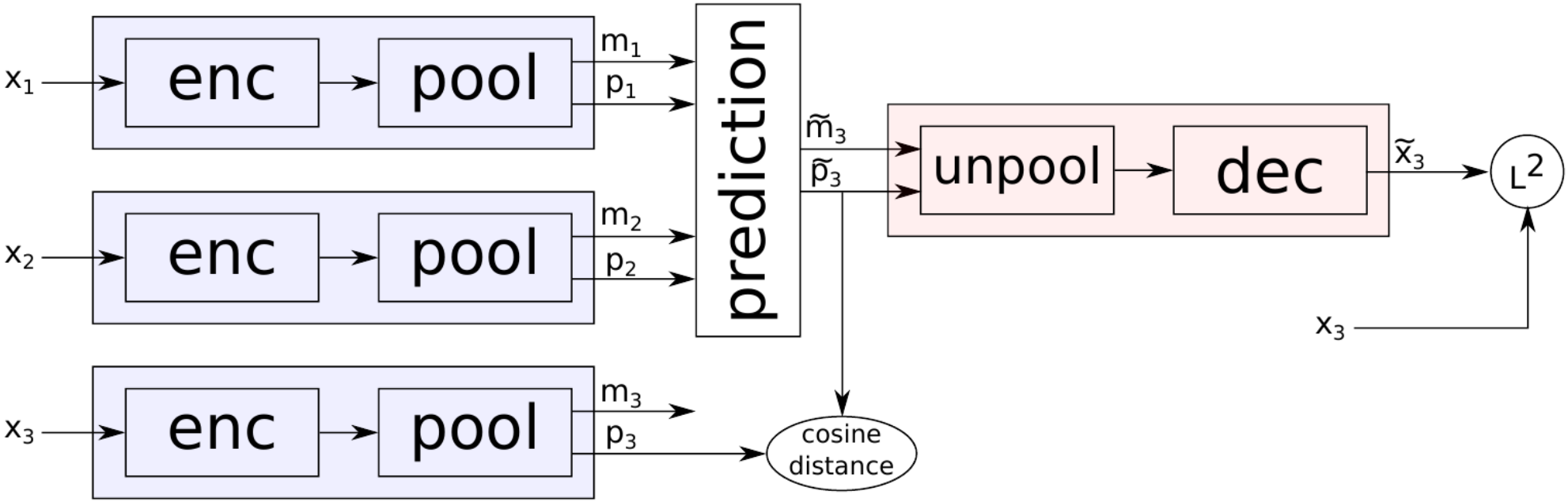
$$L = \frac{1}{2}\|G_W(\mathbf{a}\begin{bmatrix} z^t & z^{t-1}\end{bmatrix}^T) - x^{t+1}\|_2^2 - \lambda \frac{(z^t - z^{t-1})^T(z^{t+1} - z^t)}{\|z^t - z^{t-1}\|\|z^{t+1} - z^t\|}$$

- **Magnitude**
  - (soft max)

$$m_k = \sum_{N_k} z(f,x,y) \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f\prime,x\prime,y\prime)}} \approx \max_{N_k} z(f,x,y)$$

- **Phase**
  - (soft argmax)

$$\mathbf{p}_k = \sum_{N_k} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f\prime,x\prime,y\prime)}} \approx \arg\max_{N_k} z(f,x,y)$$

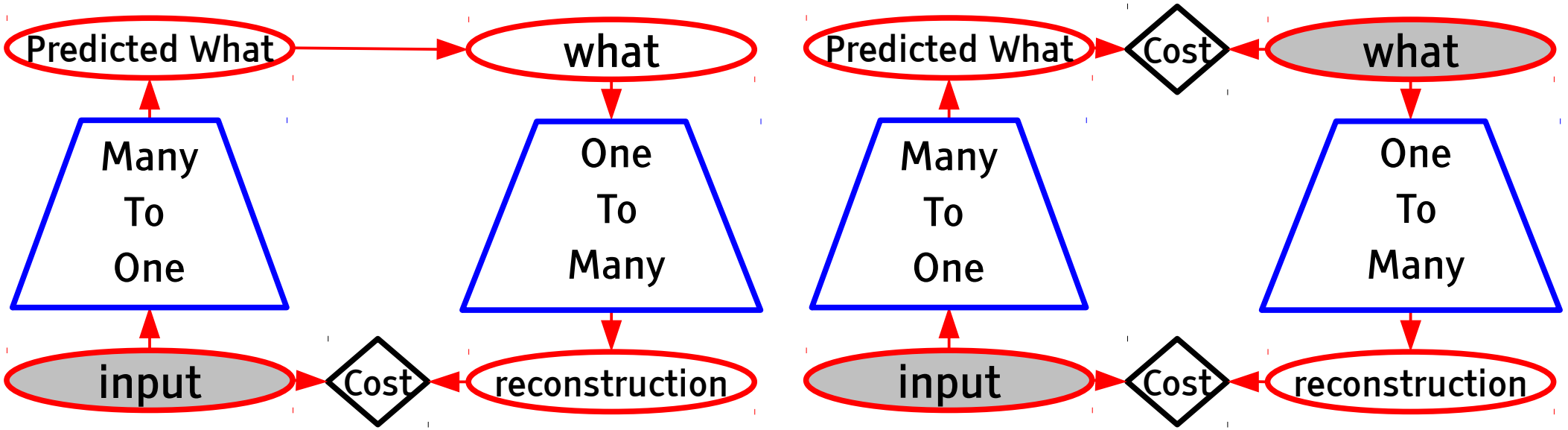



Integrating Supervised &
Unsupervised Learning
With
Stacked What-Where Auto-Encoders

Supervised and Unsupervised in One Learning Rule?

- **Boltzmann Machines have all the right properties [Hinton 1893] [OK, OK 1983 ;-]**
  - Sup & unsup, generative & discriminative in one simple/local learning rule
  - Feedback circuit reconstructs and propagates virtual hidden targets
  - But they don't really work (or at least they don't scale).
- **Problem: the feedforward path eliminates information**
- **If the feedforward path is invariant, then**

  the reconstruction path is a one-to-many mapping
  - Usual solution: sampling. But I'm allergic.



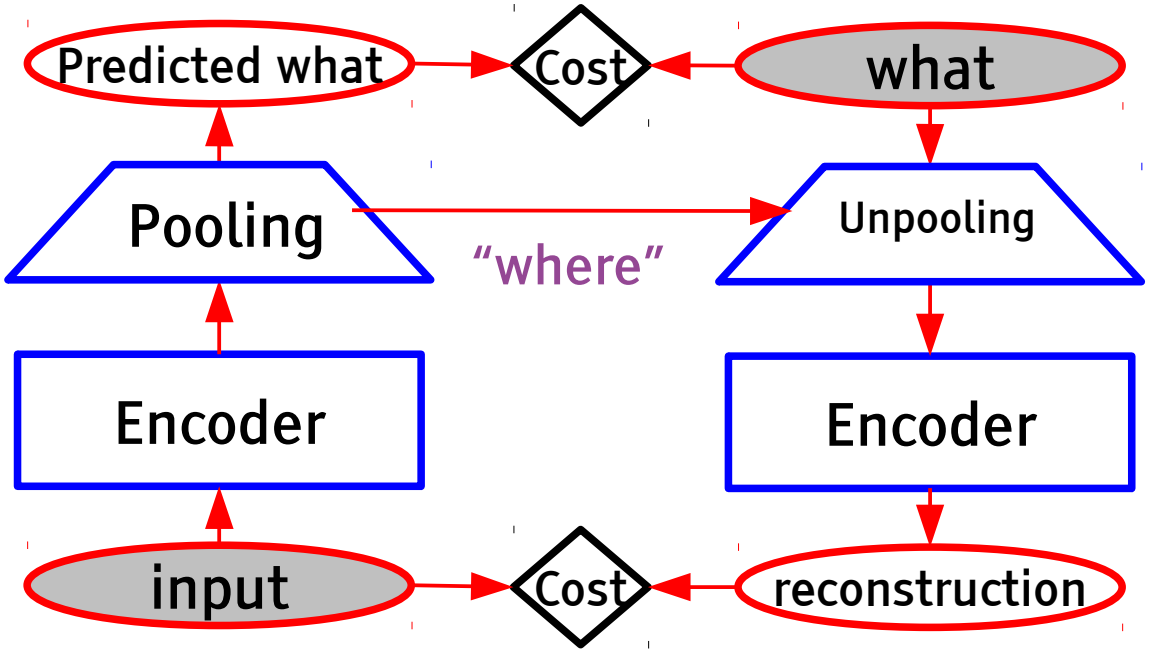
Supervised and Unsupervised in One Learning Rule?

Idea: keep the complementary information around

- So that the generative path is a function

What-Where Auto-Encoder

- [Ranzato 2007], [Gregor 2011], Hinton's Capsules
- Very old ideas by Hinton & Zemel, von der Malsburg and others about separating identity from instantiation parameters.

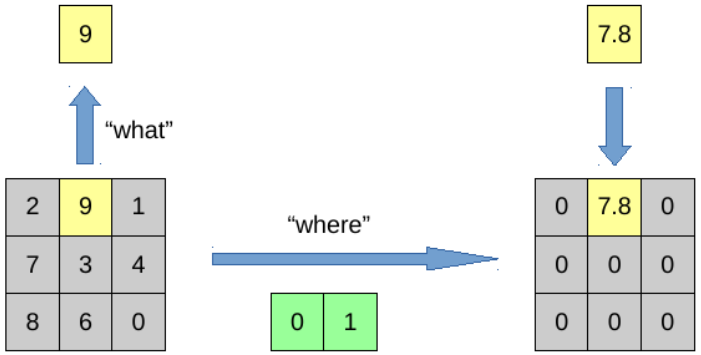


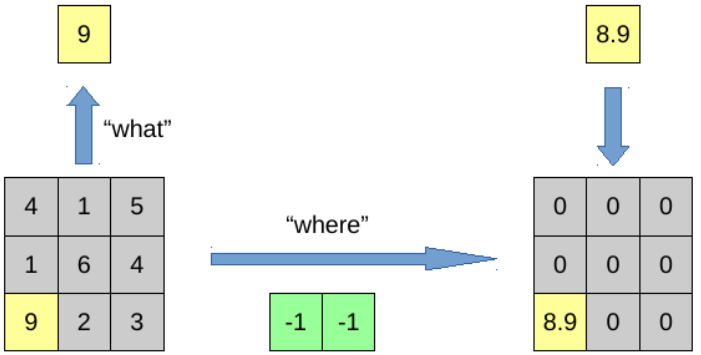
Computing the "where": Phase Pooling

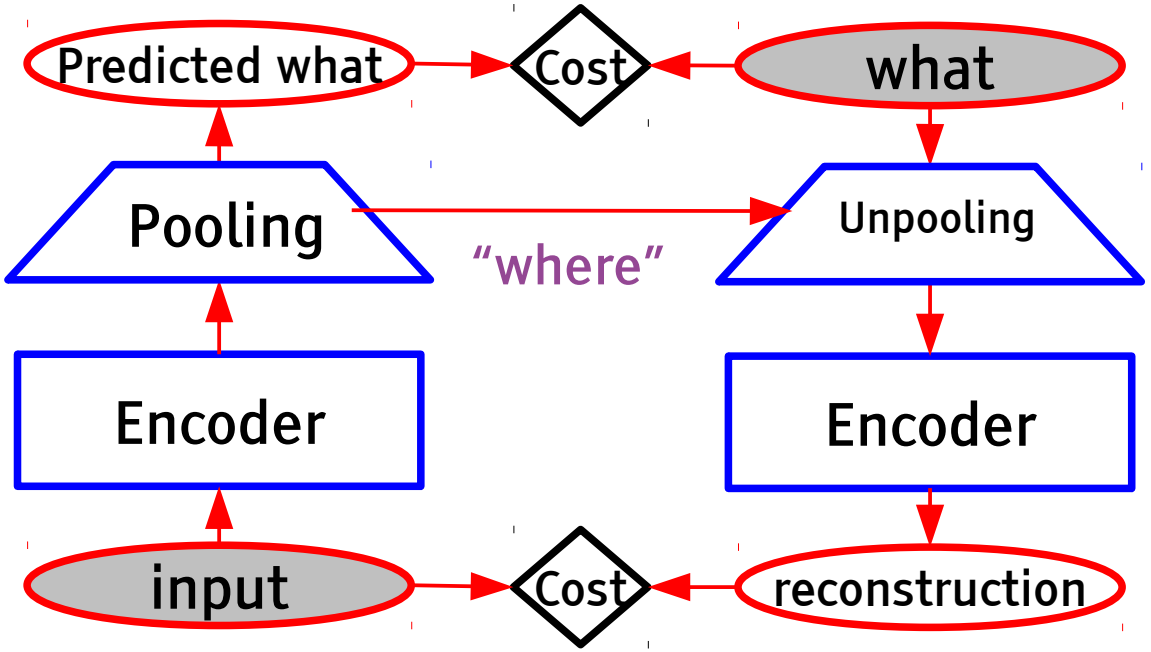
A funny kind of pooling/unpooling

$$m_k = \sum_{N_k} z(f,x,y) \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f',x',y')}} \approx \max_{N_k} z(f,x,y)$$

$$\mathbf{p}_k = \sum_{N_k} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f,x,y)}}{\sum_{N_k} e^{\beta z(f',x',y')}} \approx \arg\max_{N_k} z(f,x,y)$$



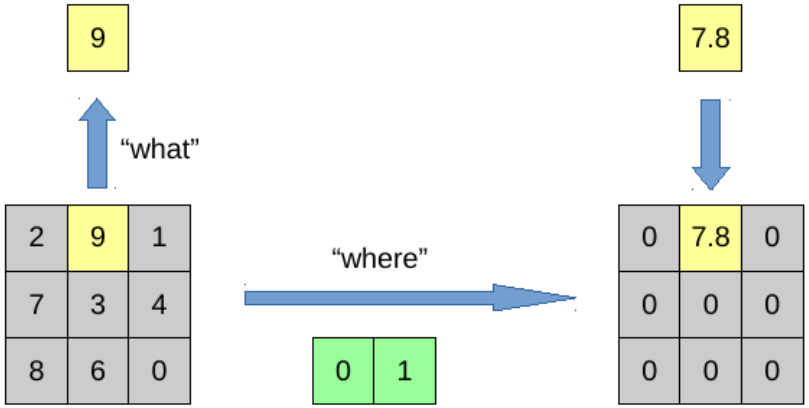




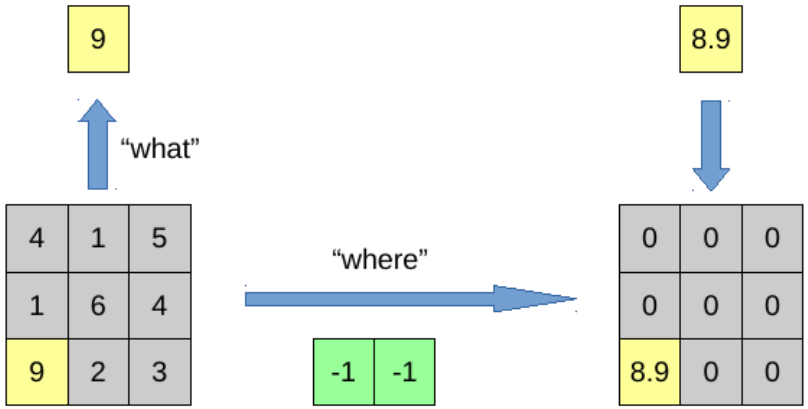
Stacked What-Where Auto-Encoder (SWWAE)

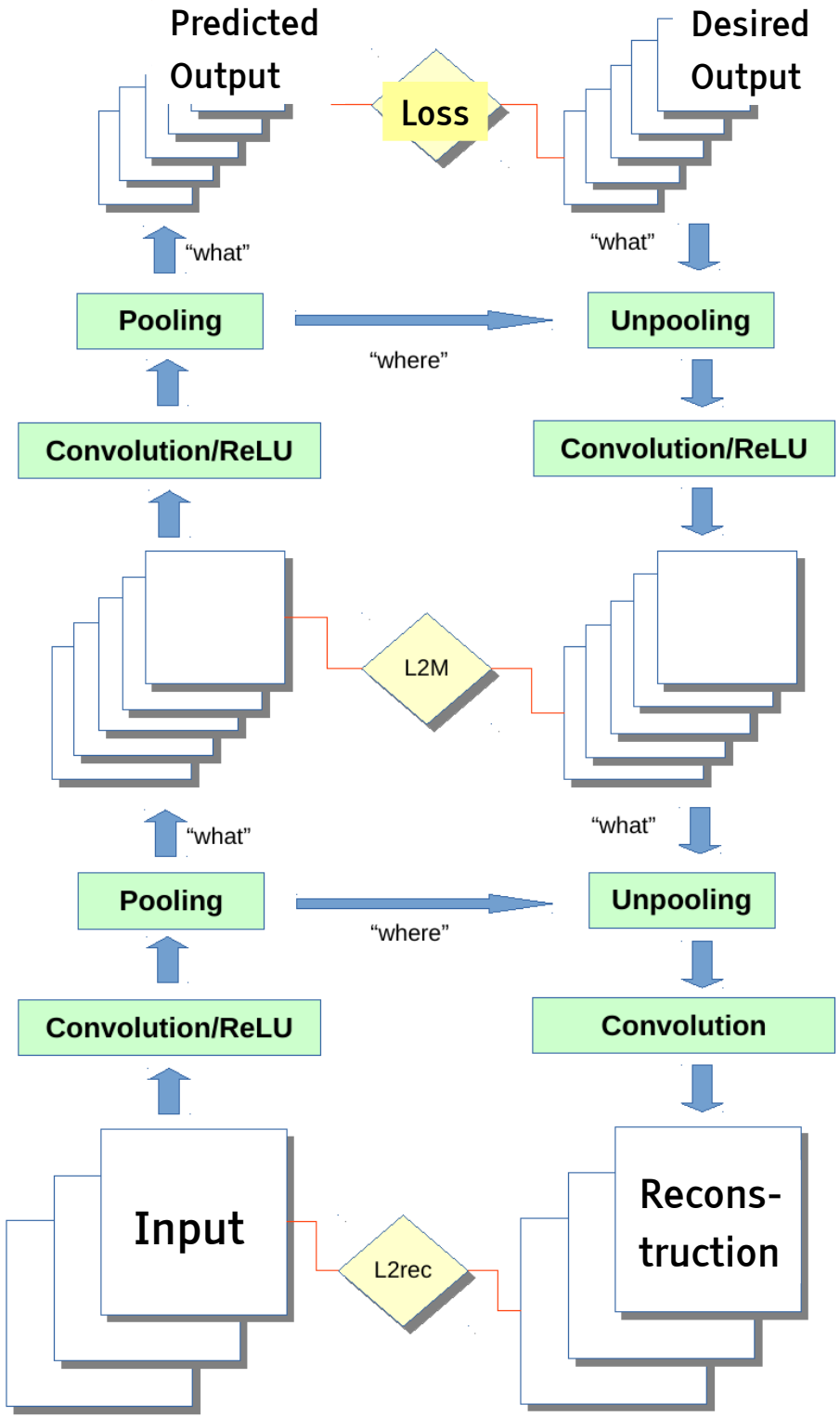
[Zhao, Mathieu, LeCun arXiv:1506.02351]

Stacked What-Where Auto-Encoder



A bit like a ConvNet paired with a DeConvNet





SWWAE: Reconstructions with Unpooling

Network: MNNIST → [Conv 5x5] → 16 fmaps → Conv 3x3 → 32 fmaps → Pooling PxP

Trained unsupervised. Hard max-pooling at test time.

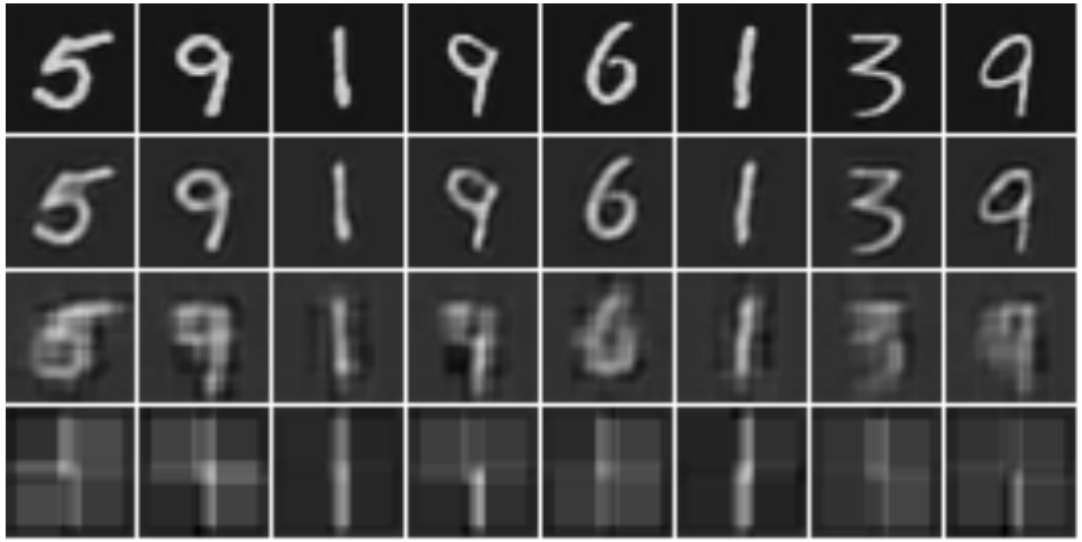
Upsampling

P=2

P=4

P=8

P=16



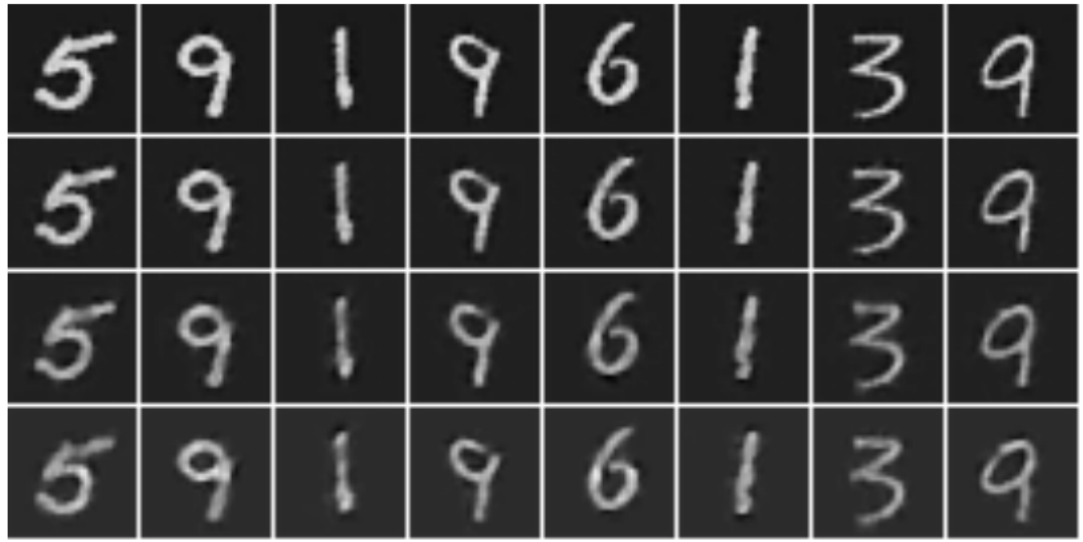
Unpooling

P=2

P=4

P=8

P=16



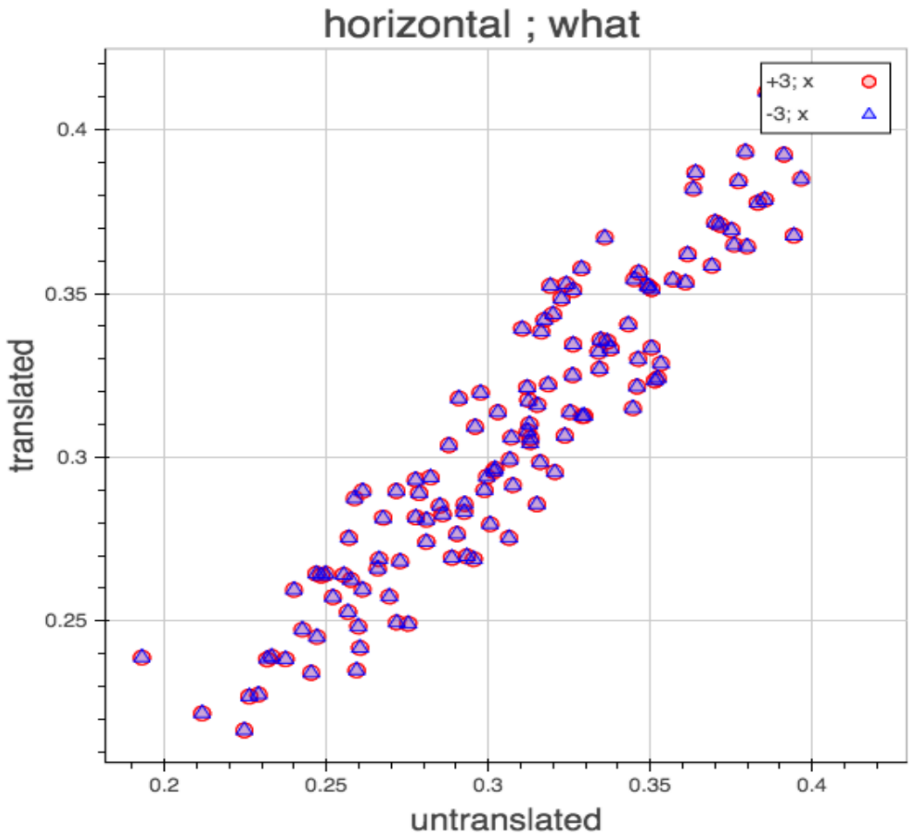
SWWAE: "What" is invariant to translations. "Where" isn't

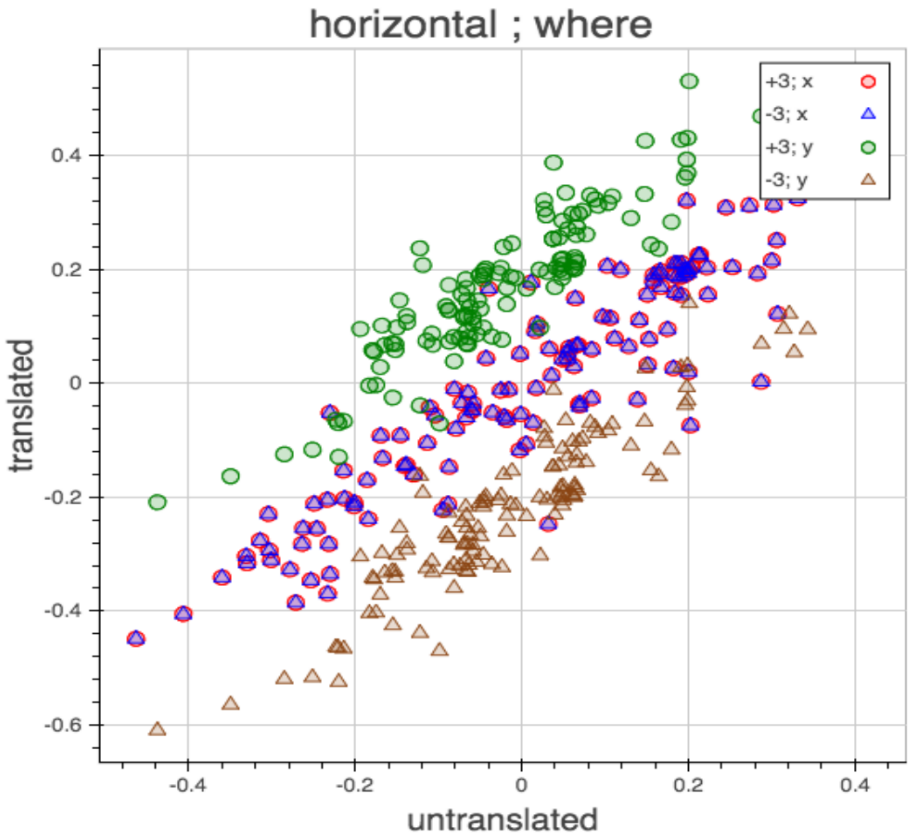
Y LeCun

Activations of features for untranslated and translated inputs

Left: "what". Activations hardly change with -3 and +3 translations

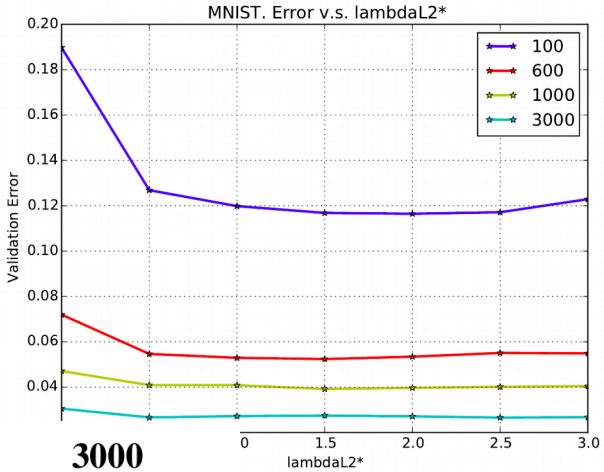
Right: "Where". Activity changes with shift.





MNIST: Recognition & Generation

model / N	100	600	1000	3000
SWWAE	[1]11.65 ± 0.20	[1]5.24 ± 0.06	[1]3.92 ± 0.09	[2]2.66 ± 0.07
dp	21.11 ± 0.65	7.11 ± 0.27	5.34 ± 0.39	3.23 ± 0.30
dp-fc	16.09 ± 0.96	6.14 ± 0.36	4.368 ± 0.50	2.98 ± 0.08
unsup-sfx	17.81 ± 0.06	8.41 ± 0.08	6.40 ± 0.06	4.76 ± 0.03
unsup-pretr	-	9.80 ± 0.06	6.135 ± 0.03	4.41 ± 3.11
noL2M	[2]13.48 ± 2.35	[2]5.69 ± 0.33	[2]3.97 ± 0.37	[1]2.52 ± 0.06



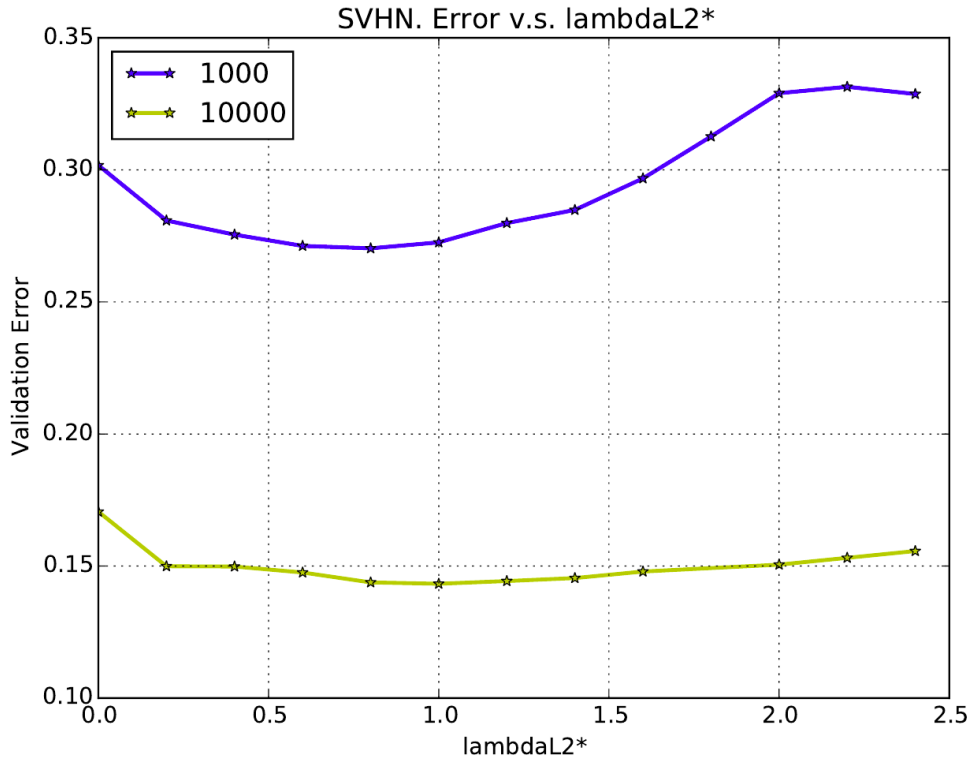
model / N	100	600	1000	3000
Convnet (LeCun et al. (1998))	22.98	7.86	6.45	3.35
TSVM (Vapnik & Vapnik (1998))	16.81	6.16	5.38	3.45
CAE (Rifai et al. (2011b))	13.47	6.3	4.77	3.22
MTC (Rifai et al. (2011a))	12.03	5.13	3.64	2.57
PL-DAE (Lee (2013))	10.49	5.03	3.46	2.69
WTA-AE (Makhzani & Frey (2014))	-	**2.37**	1.92	-
M1+M2 (Kingma et al. (2014))	3.33 ± 0.14	2.59 ± 0.05	2.40 ± 0.02	2.18 ± 0.04
LadderNetwork (Rasmus et al. (2015a))	**1.13 ± 0.04**	-	**1.00 ± 0.06**	-
SWWAE without dropout	9.17 ± 0.11	4.16 ± 0.11	3.39 ± 0.01	2.50 ± 0.01
SWWAE with dropout	8.71 ± 0.34	3.31 ± 0.40	2.83 ± 0.10	**2.10 ± 0.22**

SVHN: Recognition

- House Numbers
- Network:

(128)5c-2p-(128)3c-(256)3c-2p-(256)3c-2p

- Error rate on full dataset
  - No reconstruction: 5.89%
  - With reconstruction: 4.94%



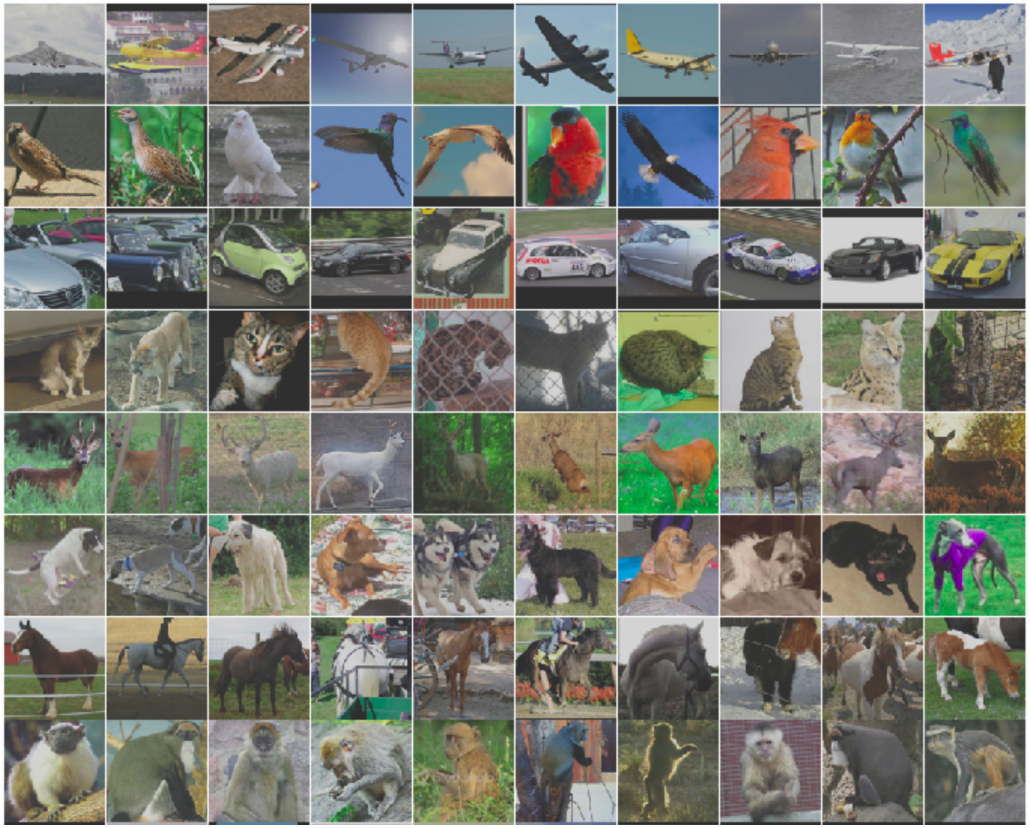
model / N	1000	10000
SWWAE	[1]27.03	[1]14.33
dp	32.25	15.72
dp-fc	[2]27.81	[2]14.39
L1	30.22	14.57
unsup-pretr	33.03	17.31
noL2M	29.12	16.51

model / N	1000 labels	error rate (in %)
KNN		77.93
TSVM		66.55
M1+KNN (Kingma et al. (2014))		65.63
M1+TSVM (Kingma et al. (2014))		54.33
M1+M2 (Kingma et al. (2014))		36.02
SWWAE without dropout ($\lambda_{L2*} = 0.8$)		27.83
SWWAE with dropout ($\lambda_{L2*} = 0.4$)		**23.56**

STL-10: Recognition

- 10 classes: airplane, bird, cat, car, deer, dog, horse, monkey, ship, truck
- 96x96 color images (from ImageNet)
- 10 predefined folds with 1000 training samples each (100 per class)
- 5000 total training samples
- 100K unlabeled samples (other categories)
- 8000 test samples (800 per class)

  [Coates et al. 2011]
- (64)3c-4p-(64)3c-3p-(128)3c-(128)3c-2p-(256)3c-(256)3c-(256)3c-(512)3c-(512)3c-(512)3c-2p-10fc

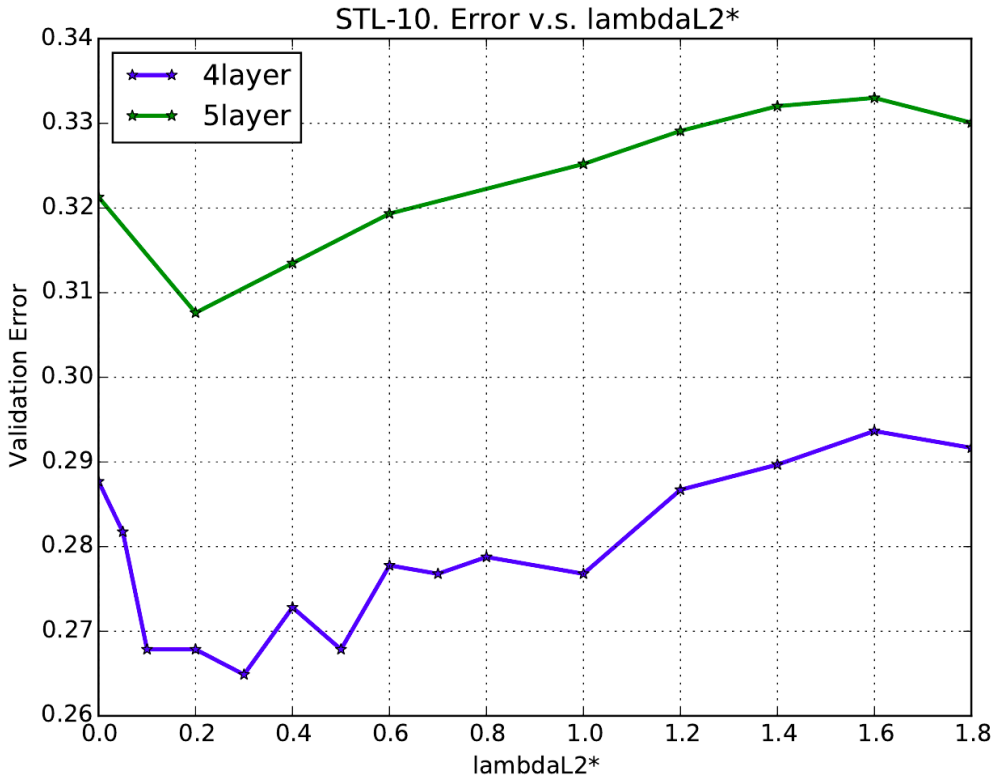


model	accuracy
Multi-task Bayesian Optimization (Swersky et al. (2013))	70.1%
Zero-bias Convnets + ADCU (Paine et al. (2014))	70.2%
Exemplar Convnets (Dosovitskiy et al. (2014))	**75.4%**
SWWAE	74.33%
Convnet of same configuration	57.45%

STL-10: Recognition

- 10 classes: airplane, bird, cat, car, deer, dog, horse, monkey, ship, truck
- 96x96 color images (from ImageNet)
- 10 predefined folds with 1000 training samples each (100 per class)
- 5000 total training samples
- 100K unlabeled samples (other categories)
- 8000 test samples (800 per class)

[Coates et al. 2011]



model	accuracy(val)
SWWAE-4layer	[1]73.51%
SWWAE-5layer	69.24%
noL2M-5layer	68.26%
noL2M-4layer	70.93%
dp-4layer	70.54%
dp-fc-4layer	71.43%

model	accuracy
Convolutional Kernel Networks [17]	62.32%
HMP [1]	64.5%
NOMP [16]	67.9%
Multi-task Bayesian Optimization [27]	70.1%
Zero-bias Convnets + ADCU [20]	70.2%
Exemplar Convnets [2]	72.8%
SWWAE-4layer	[1]74.80%

CIFAR-10, CIFAR-100, Unsup on 80M Tiny Images

- 60,000 labeled samples
- 80 Million unlabeled samples
- (128)3c-(256)3c-2p-(256)3c-(512)3c-2p-(512)3c-(512)3c-2p-(512)3c-(512)3c-2p-128fc-10fc

model	CIFAR-10	CIFAR-100
All-Convnet (Springenberg et al. (2014))	**92.75%**	66.29%
Highway Network (Srivastava et al. (2015))	92.40%	67.76%
Deeply-supervised nets (Lee et al. (2014))	92.03%	65.43%
Fractional Max-pooling with large augmentation (Graham (2014))	95.50%	68.55%
SWWAE ($\lambda_{L2rec} = 1, \lambda_{L2M} = 0.2$)	92.23%	**69.12%**
Convnet of same configuration	91.33%	67.50%